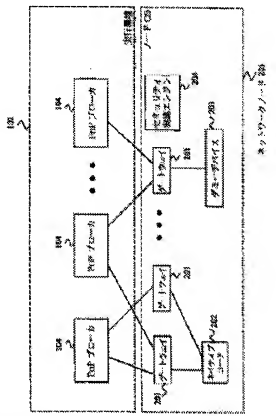
FRAMEWORK HAVING PLUG-AND-PLAY FUNCTION AND ITS RECONSTRUCTING METHOD

Publication number: JP2001290724
Publication date: 2001-10-19
Inventor: VISHAL SHAH
Applicant: NIPPON ELECTRIC CO
Classification:
- International: ***G06F13/38; G06F13/00; G06F15/00; H04L12/28; H04L29/06;*** **G06F13/38; G06F13/00; G06F15/00; H04L12/28; H04L29/06;** (IPC1-7): G06F13/00; G06F13/38; G06F15/00; H04L12/28; H04L29/06
- European:
Application number: JP20000371402 20001206
Priority number(s): US20000546397 20000410

Also published as:
JP2004334896 (/

Abstract of **JP2001290724**

PROBLEM TO BE SOLVED: To enable a user application to find and utilize various network devices by integrating a plurality of networks having different communication protocols into a single unified framework. SOLUTION: This active configuration framework system interconnects a service and a service user. The framework has a plug-and-play(PnP) broker 104. The service user finds a networking service by using the PnP broker, utilizes the networking service and communicates with the networking service. The service user communicates with the service through the interface of the PnP broker independently of a communication protocol used by a device offering the service. A gateway 201 for registering the service in the PnP broker is also provided.



(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号
特開2001－290724
（P2001－290724A）
(43)公開日　平成13年10月19日(2001.10.19)

(51)Int.Cl.[7]	識別記号	Ｆ Ｉ		ﾃｰﾏｺｰﾄﾞ(参考)
Ｇ０６Ｆ　13/00	３５７	Ｇ０６Ｆ	13/00	３５７Ａ
	３５１			３５１Ｂ
13/38	３５０		13/38	３５０
15/00	３１０		15/00	３１０Ｄ
Ｈ０４Ｌ　12/28		Ｈ０４Ｌ	11/00	３１０Ｚ

審査請求 有　請求項の数32 ＯＬ （全 24 頁）　最終頁に続く

(21)出願番号　特願2000－371402(P2000－371402)

(22)出願日　平成12年12月６日(2000.12.6)

(31)優先権主張番号　０９／５４６３９７

(32)優先日　平成12年４月10日(2000.4.10)

(33)優先権主張国　米国（ＵＳ）

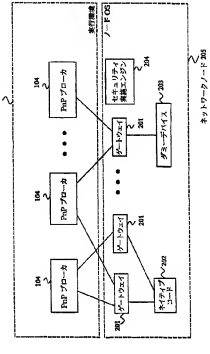
(71)出願人　000004237
日本電気株式会社
東京都港区芝五丁目７番１号

(72)発明者　ビシャル・シャー
アメリカ合衆国、カリフォルニア　95134
サンノゼ、110　ロブルスアベニュー、
エヌ・イー・シー・ユー・エス・エー・イ
ンク内

(74)代理人　100097157
弁理士　桂木　雄二

(54)【発明の名称】　プラグアンドプレイ機能を有するフレームワークおよびその再構成方法

(57)【要約】
【課題】　相異なる通信プロトコルを有する複数のネットワークを単一の統一フレームワークへと統合して、ユーザアプリケーションがさまざまなネットワークデバイスを発見し利用することを可能にする。
【解決手段】　本発明のアクティブコンフィグレーションフレームワークシステムは、サービスとサービスユーザを相互接続する。このフレームワークは、プラグアンドプレイ（ＰｎＰ）ブローカ１０４を有する。サービスユーザは、ＰｎＰブローカを用いて、ネットワーキングサービスを発見し、利用し、ネットワーキングサービスと通信する。サービスユーザは、サービスを提供するデバイスによって使用される通信プロトコルとは独立に、ＰｎＰブローカのインタフェースを通じてサービスと通信する。また、サービスをＰｎＰブローカに登録するゲートウェイ２０１が提供される。



1

【特許請求の範囲】

【請求項１】　（ａ）少なくとも１つのサービスおよび少なくとも１つのサービスユーザと、

（ｂ）第１のプラグアンドプレイブローカとを有するアクティブコンフィグレーションフレームワークにおいて、

前記少なくとも１つのサービスユーザは、前記第１のプラグアンドプレイブローカを用いて、前記サービスを利用することを特徴とするアクティブコンフィグレーションフレームワーク。

【請求項２】　前記サービスユーザは、前記サービスを提供するデバイスによって使用される通信プロトコルとは独立に、前記第１のプラグアンドプレイブローカを通じて前記サービスと通信することを特徴とする請求項１に記載のアクティブコンフィグレーションフレームワーク。

【請求項３】　前記サービスは、ゲートウェイおよびデバイスクラスタを有し、

前記ゲートウェイは、前記デバイスクラスタを前記第１のプラグアンドプレイブローカに接続し、

前記第１のプラグアンドプレイブローカは、前記ゲートウェイを通じて、前記デバイスクラスタと通信することを特徴とする請求項１に記載のアクティブコンフィグレーションフレームワーク。

【請求項４】　前記第１のプラグアンドプレイブローカは、第１の通信プロトコルを用いて前記ゲートウェイに接続され、

前記ゲートウェイは、異なる第２の通信プロトコルを用いて前記デバイスクラスタに接続され、

前記ゲートウェイは、前記第１の通信プロトコルと前記第２の通信プロトコルの間の変換を行うことを特徴とする請求項３に記載のアクティブコンフィグレーションフレームワーク。

【請求項５】　前記ゲートウェイは、前記第１のプラグアンドプレイブローカ内の、前記サービスに対応する属性を設定することによって、前記サービスを前記第１のプラグアンドプレイブローカに登録することを特徴とする請求項３に記載のアクティブコンフィグレーションフレームワーク。

【請求項６】　前記ゲートウェイは、前記第１のプラグアンドプレイブローカからの要求に応答して、前記第１のプラグアンドプレイブローカで前記サービスのステータスを周期的に更新することを特徴とする請求項３に記載のアクティブコンフィグレーションフレームワーク。

【請求項７】　前記サービスユーザは、前記第１のプラグアンドプレイブローカから離れたリモートサービスユーザであり、

前記リモートサービスユーザは、インターネット通信プロトコルを通じて前記第１のプラグアンドプレイブローカと通信することを特徴とする請求項１に記載のアクティブコンフィグレーションフレームワーク。

2

【請求項８】　前記アクティブコンフィグレーションフレームワークは、第２のプラグアンドプレイブローカをさらに有し、

前記サービスユーザは、前記第２のプラグアンドプレイブローカと通信し、

前記第２のプラグアンドプレイブローカは、前記第１のプラグアンドプレイブローカと通信して、前記少なくとも１つのサービスユーザが前記サービスを利用することを可能にすることを特徴とする請求項１に記載のアクティブコンフィグレーションフレームワーク。

【請求項９】　前記アクティブコンフィグレーションフレームワークは、ディレクトリエージェントをさらに有し、

前記第１のプラグアンドプレイブローカは、サービスロケーションプロトコルを用いて、前記サービスを前記ディレクトリエージェントに登録することを特徴とする請求項８に記載のアクティブコンフィグレーションフレームワーク。

【請求項１０】　前記第１のプラグアンドプレイブローカは、サービスレジストリを有し、

前記サービスレジストリは、前記サービスおよび他のネットワークサービスに関する情報を含み、

前記少なくとも１つのサービスユーザは、前記サービスレジストリを用いて、前記サービスおよび前記他のネットワークサービスを発見することを特徴とする請求項１に記載のアクティブコンフィグレーションフレームワーク。

【請求項１１】　前記アクティブコンフィグレーションフレームワークは、第２のプラグアンドプレイブローカをさらに有し、

前記第２のプラグアンドプレイブローカは、前記第１のプラグアンドプレイブローカに問合せを送ることによって、前記サービスを発見することを特徴とする請求項１に記載のアクティブコンフィグレーションフレームワーク。

【請求項１２】　前記アクティブコンフィグレーションフレームワークは、第２のプラグアンドプレイブローカおよびディレクトリエージェントをさらに有し、

（ａ）前記少なくとも１つのサービスユーザは、要求されるサービスの記述を前記第２のプラグアンドプレイブローカに送り、

（ｂ）前記第２のプラグアンドプレイブローカは、前記ディレクトリエージェントと通信し、

（ｃ）前記ディレクトリエージェントは、前記要求されるサービスの記述に一致するサービスのアドレスを前記第２のプラグアンドプレイブローカに返し、

（ｄ）前記第２のプラグアンドプレイブローカは、前記アドレスを用いて、前記サービスユーザと前記サービスの間にセッションを確立することを特徴とする請求項１

に記載のアクティブコンフィグレーションフレームワーク。

【請求項１３】　（１）前記サービスは、ゲートウェイおよびデバイスクラスタを有し、

（２）前記ゲートウェイは、前記デバイスクラスタを前記第１のプラグアンドプレイブローカに接続し、

（３）前記第１のプラグアンドプレイブローカは、前記ゲートウェイを通じて、前記デバイスクラスタと通信し、

（４）前記ゲートウェイは、前記第１のプラグアンドプレイブローカ内の、前記サービスに対応する属性を設定することによって、前記サービスを前記第１のプラグアンドプレイブローカに登録し、

（５）前記第１のプラグアンドプレイブローカは、前記サービスを前記ディレクトリエージェントに登録することを特徴とする請求項１０に記載のアクティブコンフィグレーションフレームワーク。

【請求項１４】　前記第１のプラグアンドプレイブローカは、

（ａ）前記サービスユーザと通信するプラグアンドプレイブローカインタフェースと、

（ｂ）前記第１のプラグアンドプレイブローカと、前記フレームワーク内の他のプラグアンドプレイブローカとの間の通信のためのプラグアンドプレイブローカ間プロトコルと、

（ｃ）前記サービスに関する情報を格納する少なくとも１つのサービスレコードを有するサービスレジストリと、

（ｄ）前記サービスのネットワークアドレスを探索し前記サービスのアベイラビリティステータスを取得するサービスディスカバリ／アベイラビリティエージェントと、

（ｅ）前記サービスと前記サービスユーザの間の通信セッションを確立するサービスセッション管理エージェントと、

（ｆ）前記サービスのアドレスを探索するサービスロケーションプロトコルユーザエージェントと、

（ｇ）前記サービスを公表するサービスロケーションプロトコルサービスエージェントとを有することを特徴とする請求項１に記載のアクティブコンフィグレーションフレームワーク。

【請求項１５】　前記少なくとも１つのサービスレコードは、前記サービスのタイプおよび複数のサービス属性レコードを格納するサービス識別フィールドを有し、

前記サービス属性レコードは、前記サービス属性のタイプを格納する属性識別フィールドと、前記サービス属性の値を格納する属性値フィールドと、前記サービスのアクセス制御情報を格納するアクセス制御フィールドとを有することを特徴とする請求項１４に記載のアクティブコンフィグレーションフレームワーク。

【請求項１６】　（ａ）少なくとも１つのサービスと、

（ｂ）サービスエージェントと、

（ｃ）ディレクトリエージェントとを有するアクティブコンフィグレーションフレームワークにおいて、

前記サービスエージェントは、前記少なくとも１つのサービスのタイプおよび属性を前記ディレクトリエージェントに登録し、

前記ディレクトリエージェントは、ユーザアプリケーションによって要求されるサービスのタイプおよび属性を、前記少なくとも１つのサービスのタイプおよび属性と照合して、前記少なくとも１つのサービスのアドレスを返すことを特徴とするアクティブコンフィグレーションフレームワーク。

【請求項１７】　前記アクティブコンフィグレーションフレームワークは、ユーザエージェントをさらに有し、

前記ユーザエージェントは、前記要求されるサービスのタイプおよび属性を前記ディレクトリエージェントに送信し、

前記ディレクトリエージェントは、前記少なくとも１つのサービスのアドレスを前記ユーザエージェントに返すことを特徴とする請求項１６に記載のアクティブコンフィグレーションフレームワーク。

【請求項１８】　前記少なくとも１つのサービスは、ゲートウェイおよびデバイスクラスタを有し、

前記ゲートウェイは、前記デバイスクラスタを前記サービスエージェントに接続し、

前記サービスエージェントは、前記ゲートウェイを通じて、前記デバイスクラスタと通信することを特徴とする請求項１６に記載のアクティブコンフィグレーションフレームワーク。

【請求項１９】　前記アクティブコンフィグレーションフレームワークは、実行環境をさらに有し、

前記アプリケーションは、前記実行環境で動作することを特徴とする請求項１６に記載のアクティブコンフィグレーションフレームワーク。

【請求項２０】　前記実行環境はＪａｖａ仮想マシンであることを特徴とする請求項１９に記載のアクティブコンフィグレーションフレームワーク。

【請求項２１】　前記実行環境はＷｉｎｄｏｗｓ™ベースのマシンであることを特徴とする請求項１９に記載のアクティブコンフィグレーションフレームワーク。

【請求項２２】　ネットワークデバイスの追加に応答してフレームワークの自動再構成を行う方法において、

前記ネットワークデバイスは、少なくとも１つのサービスを提供し、

前記フレームワークは、サービスエージェントおよびディレクトリエージェントを有し、

前記方法は、

（ａ）前記サービスエージェントが、提供されるサービスのタイプおよび属性を前記ディレクトリエージェント

に登録するステップと、
（ｂ）前記ディレクトリエージェントが、ユーザアプリケーションによって要求されるタイプおよび属性を、前記提供されるサービスのタイプおよび属性と照合して、前記提供されるサービスのアドレスを返すステップと、
を有することを特徴とする、ネットワークデバイスの追加に応答してフレームワークの自動再構成を行う方法。
【請求項２３】　前記フレームワークは、ユーザエージェントをさらに有し、
前記ユーザエージェントは、前記要求されるサービスのタイプおよび属性を前記ディレクトリエージェントに送信し、
前記ディレクトリエージェントは、前記提供されるサービスのアドレスを前記ユーザエージェントに返すことを特徴とする請求項２２に記載の方法。
【請求項２４】　前記ステップ（ａ）は、
（１）前記ゲートウェイが、前記ネットワークデバイスのアベイラビリティをチェックするステップと、
（２）前記ゲートウェイが、前記提供されるサービスのタイプおよび属性を前記サービスエージェントに登録するステップと、
（３）前記サービスエージェントが、前記提供されるサービスのタイプおよび属性を前記ディレクトリエージェントに登録するステップと、
を有することを特徴とする請求項２２に記載の方法。
【請求項２５】　（ｃ）セッション管理エージェントが、前記アドレスを用いて、前記ユーザアプリケーションと前記ネットワークデバイスの間の通信セッションを確立するステップをさらに有することを特徴とする請求項２２に記載の方法。
【請求項２６】　フレームワークに挿入されたデバイスによって実行されるサービスのための通信インタフェースを生成する方法において、
前記インタフェースは、ユーザアプリケーションによって、前記サービスを利用するために使用され、
前記方法は、
（ａ）ＸＭＬ文書、文書型定義、またはスタイルシートのうちの少なくとも１つを含む、前記デバイスによって提供されるサービスのサービス記述スキーマを生成するステップと、
（ｂ）ＸＭＬパーサを用いて、前記サービス記述スキーマから、前記サービスのための通信インタフェースを生成するステップと、
を有することを特徴とする、フレームワークに挿入されたデバイスによって実行されるサービスのための通信インタフェースを生成する方法。
【請求項２７】　前記サービス記述スキーマは、前記サービスのためのデータおよび制御フロー仕様を含むことを特徴とする請求項２６に記載の方法。
【請求項２８】　前記ステップ（ｂ）において、前記デバイスによって生成されるサービス記述スキーマは、ユーザインタフェースウィジェットのセットにマッピングされ、
ユーザは、実行のために前記デバイスにコマンドを送るユーザインタフェースウィジェットをクリックすることによって、前記デバイスの状態を変えることができることを特徴とする請求項２６に記載の方法。

【請求項２９】　前記ステップ（ｂ）において、前記デバイスによって生成されるサービス記述スキーマは、前記サービスのための通信インタフェースを表すオブジェクトの階層にマッピングされ、
前記サービスは、前記オブジェクトの階層を用いて、前記ユーザアプリケーションによって利用されることを特徴とする請求項２６に記載の方法。
【請求項３０】　前記オブジェクトの階層の構造は、前記ユーザアプリケーションが前記デバイスを利用する時点での前記デバイスの機能に基づいて決定されることを特徴とする請求項２９に記載の方法。
【請求項３１】　前記ＸＭＬパーサは、ＸＭＬ対応ウェブブラウザであることを特徴とする請求項３０に記載の方法。
【請求項３２】　フレームワークに挿入されたデバイスを制御する方法において、該方法は、
（ａ）前記デバイスが、ＸＭＬ文書、文書型定義、またはスタイルシートのうちの少なくとも１つを含む、前記デバイスのデバイス記述スキーマを生成するステップと、
（ｂ）前記ユーザアプリケーションが、前記デバイス記述スキーマを編集し、編集されたデバイス記述スキーマを前記デバイスに反映するステップとを有し、
前記デバイスは、前記編集されたデバイス記述スキーマに従って状態を変更することを特徴とする、フレームワークに挿入されたデバイスを制御する方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、相異なる通信プロトコルを有する複数のネットワークを単一の統一フレームワークへと統合して、ユーザアプリケーションがさまざまなネットワークデバイスを発見し利用することを可能にする方法および装置に関する。また、本発明は、利用可能な機能の変更およびユーザ相互作用に応答してネットワークノードの動的な適応および再構成を達成する方法に関する。また、本発明は、アプリケーション、サービス、およびデバイスが、自己の能力を記述し、それを他のアプリケーション、サービス、およびデバイスに公表する方法に関する。さらに、本発明は、物理的に相異なるデバイスが、接続し、情報交換し、データタイプを交渉し、それぞれの動作についてのステータスを提供して、ネットワークプラグアンドプレイを達成することを可能にする、プラットフォーム独立でトランスポート

独立なプロトコルを提供する方法に関する。

【０００２】

【従来の技術】世界的なネットワーキング基盤の出現により、分散サービスおよび分散アプリケーションの大規模な配備が一般的になっている。現在および将来のネットワークでは、時間に敏感な情報を全世界に発信し、世界的な取引を電子的に仲介するサービスがさらに配備されることが期待される。このような分散サービスが普遍的になる前に、このようなサービスの開発、デバッグ、配備および発展を簡単にする新たな機構が必要とされる。このような機構は、基礎になるプロトコルやインタフェースにかかわらずに、必要なサービスを発見し、そのサービスの能力を使用することが必要となる。

【０００３】家庭で使用可能なもののような典型的な分散ネットワークシステムは、相異なるさまざまな機器を相互接続する。さまざまな家庭用機器（ホームデバイス）は本質的にさまざまなインタフェースおよびプロトコルを利用することがあるため、単一の共通の通信プロトコルを用いてこれらのすべてのデバイスを相互接続することはほとんど不可能である。場合によっては、ほとんどの家庭に既に設置されている配線を使用する（例えば、既存の電源線をＸ－１０デバイスとの通信に使用する）ことは便利であるが、このような既存の配線を使用するネットワークの伝送速度が制限される。例えば、白熱電球やホームセキュリティシステムのようなＸ－１０デバイスは、毎秒数ビットのデータレートで動作する。他方、ディジタルＴＶ、カメラ、あるいはＶＣＲのようなデバイスは、ずっと高い伝送データレートを必要とする。このようなデバイスは通常、ＩＥＥＥ１３９４アーキテクチャプロトコルを用いて相互接続され、データ転送レートは４００Ｍｂ／ｓに達する。

【０００４】これに対して、現在開発されているさまざまな家庭用自動化アプリケーションは、さまざまな家庭用機器の有効な相互作用を必要とする。したがって、家庭に設置されたすべての相異なるデバイスおよびサービスの有効な相互作用を可能にするには、相異なるさまざまなネットワークを、統一するフレームワークに統合して、相異なるネットワーク内に位置するさまざまなネットワークエンティティが互いを発見し相互作用する機構を提供しなければならない。

【０００５】また、さまざまなデバイスインタフェースおよびネットワークプロトコルについて、低レベルのプロトコル固有の詳細をアプリケーションが扱うことを必要とせずに、リモートサービスを利用する手段をユーザアプリケーションに提供することが望ましい。

【０００６】真のプラグアンドプレイ能力を達成するには、フレームワークは、新たに追加されたデバイスによって提供されるサービスを自動的に発見する機構を提供しなければならない。このようなフレームワーク内では、利用可能なネットワークサービスおよびそのネットワーク位置に関する記述は、すべてのユーザアプリケーションにとって容易に利用可能であるようにしなければならない。さらに、ユーザアプリケーションは、遭遇する可能性のある特定のデバイスあるいはサービスとの通信のために、自己の通信インタフェースを自動的に再構成することができなければならない。さらに、フレームワークは、ネットワークサービスへのアクセスを制御するために、ユーザの認可および認証を実行するセキュリティモデルを含まなければならない。

【０００７】Ｊｉｎｉ™は、ネットワークプラグアンドプレイを達成する既存の候補のうちの１つである（Ｊｉｎｉは、サンマイクロシステムズ社(Sun Microsystems, Inc.)の商標である）。Ｊｉｎｉは、分散システムの構築および配備を容易にするアプリケーションプログラミングインタフェース（ＡＰＩ）およびランタイム規約のセットである。Ｊｉｎｉは、分散システムの、共通しているが相異なる部分を処理する「配管」(plumbing)を提供する。Ｊｉｎｉは、プログラミングモデルおよびランタイムインフラストラクチャからなる。リース、分散イベント、および分散トランザクションをサポートするＡＰＩおよび規約を定義することによって、プログラミングモデルは、基礎となるネットワークの信頼性が低くても、信頼性の高い分散システムを開発者が構築するのを助ける。ランタイムインフラストラクチャは、ネットワークプロトコルとそれを実装するＡＰＩとからなり、ネットワーク上のデバイスおよびサービスの追加、探索、アクセス、および削除を容易にする。

【０００８】Ｊｉｎｉの使用は、基礎となるネットワーク技術について知らなくてもサービスを発見する容易な方法を提供する。他の既知のインフラストラクチャに比べてＪｉｎｉが改良されている点は、ユーザアプリケーションが実際にサービスを探索し、そのサービスをサポートするホストを他者に発見させることができることである。しかし、主な未解決の問題点は、「ユーザアプリケーションはどのようにして自動的にクライアントマシン上のサービスを使用するか」である。クライアントユーザアプリケーションにサービスインタフェースしか教えられていない場合、クライアントが、このサービスを理解することが可能なコードを有していなければならない。

【０００９】例えば、ユーザが、自己のアプリケーションに、カメラサービスのためのネットワークをブラウズさせたい状況を考える。アプリケーションがこのようなサービスをネットワーク上に見つけた場合、ユーザは、それをクリックして、ユーザ自身のアプリケーションでそれを使用することができる。ユーザが、このようなサービスを実行するためのコードを必要とする場合、このようなコードは、ユーザのマシン上に自動的にダウンロードされなければならない。しかし、このアプローチに関連する主な問題点は、ユーザアプリケーションがこの

9

ようなサービスをどのようにして利用するかである。ユーザアプリケーションは、拡張可能なインタフェースを有する場合、リモートサービスインタフェースに問い合わせてその固有の特性を取得し、その利用を可能にすることができる。しかし、このような拡張可能インタフェースの定義は、Ｊｉｎｉ™仕様の範囲内にはなく、Ｊｉｎｉ™サービスを使用するユーザアプリケーションに委ねられている。

【００１０】もう１つの問題点は、カメラサービスのセマンティクス（意味論）を定義するエンティティの識別である。例えば、キャノン、ミノルタ、およびニコンを含む主要なカメラメーカがそれぞれ固有のインタフェースを提供する場合、カメラサービスには相互運用性がなくなり、ユーザは、３つの異なるすべてのカメラサービスをコードの形で適切に提供しなければならないことになる。また、もしコダックがＪｉｎｉ™カメラを市販することに決めた場合、ユーザは、コダックのインタフェースの解析を含むようにそのコードを手動（マニュアル）で変更しなければならない。このような拡張可能インタフェースは、ネットワークトラフィックと、クライアントコードのサイズを非常に増大させてしまう。すべての種類の市場固有のグループがまとまってインタフェースを指定することが好ましく、多くの場合にはこれが必要となるであろうが、これはすぐには、また効率的には実現しそうになく、特に、家庭で使用されるすべての種類のデバイスに対しては、きわめて非現実的である。

【００１１】既知のネットワークで依然として解決されていないもう１つの制限は、Ｊｉｎｉ™ベースのアプリケーションがどのようにして非Ｊｉｎｉ™ベースのオペレーティングシステム上にあるデータにアクセスすることができるかに関するものである。すなわち、ユーザは、例えば、Ｊｉｎｉ™に対するマイクロソフトのサポートなしで、どのようにしてＭＳ－Ｗｏｒｄ™からＪｉｎｉ™プリンタにアクセスすることができるだろうか。Ｊｉｎｉプリンタを使用したいＪａｖａ™アプリケーションですら、アクセスのためのオペレーティングシステムに依拠しなければならない。

【００１２】例えば、ユーザプログラムが印刷メソッドへのアクセスを要求するとき、以下のようなコードフラグメントが使用される。

```
object o=Lookup(GeneralPrintService);
GeneralPrintService pservice=(GeneralPrintService)o;
pservice.print();
```

【００１３】しかし、このコードが書かれるときに、プログラマは、GeneralPrintServiceという名前のインタフェースがコード実行時に本当に存在するかどうかを知らない。さらに、プログラマが複数のデバイスでイメージを印刷したい場合、複数のインタフェースを予想し、それに応じて印刷を行う必要がある。

10

【００１４】Ｊｉｎｉは、ユーザインタフェース（ＵＩ）コンポーネントがサービスの属性リストにエントリとして付加されるように使用することができるため、サービスについて要求されるクライアントの知識は最小限である。クライアントは、単に、ＵＩがアプレットと同様の属性によって表現されることを知っているだけでよい。ＵＩは、ＪａｖａＢｅａｎｓとして提供可能であるため、内省（introspection）・反省（reflection）により、サービスは、クライアントに対して、クライアントがそのサービスを使用することができる手段を提供することが可能である。これは、本質的に、クライアントとサーバの間のメッセージ交換であり、手動の設定を必要とする。しかし、ＵＩコンポーネントを付加することは、人間でないクライアント（生成・使用されるマシン）がデバイスを使用するのを妨げる可能性がある。また、デバイスの個数が増大して、クライアントプログラムが複雑になると、もう１つの問題点が生じる。その場合、クライアントは、返されたリストのどのオブジェクトが真のターゲットであるかを決定しなければならない。さらに、あらゆるＪｉｎｉ™のデバイスあるいはアプリケーションは、製造時において通信するために必要としたすべてのデバイスの（あるいは少なくとも、「デバイスのすべてのタイプの」）すべてのインタフェースを必要とする。その結果、Ｊｉｎｉ™は、相互運用性の問題点を実際には解決していない。

【００１５】もう１つの既存のネットワーキングシステムであるＨｏｍｅＡＰＩは、アプリケーションがさまざまなホームデバイスを発見し利用することを可能にする、オブジェクト指向のソフトウェアサービスおよびアプリケーションプログラミングインタフェース（ＡＰＩ）のセットである。このシステムは、アプリケーションが使用するデバイスおよびネットワークのプロトコル固有の詳細に、ユーザアプリケーションが関与しないように設計された。ＨｏｍｅＡＰＩは、ＨｏｍｅＡＰＩインタフェースの使用を通じてサービスを利用するユーザアプリケーションを開発するのを容易にする分散サービスを表現するソフトウェアオブジェクトのセットを提供する。

【００１６】しかし、ＨｏｍｅＡＰＩは、新たなデバイスの発見の問題点を解決していないため、所望のプラグアンドプレイネットワーキング能力を提供しない。ＨｏｍｅＡＰＩのもう１つの欠点は、すべてのネットワークデバイスが標準の通信インタフェースを提供することを要求することであり、これは今日の市場では実現不可能である。

【００１７】ＨｏｍｅＰＮＡ、ＨｏｍｅＲＦ、Ｂｌｕｅｔｏｏｔｈ、ＰＩＡＮＯ、Ｘ－１０、ＣＥＢｕｓ、ＩＥＥＥ１３９４、ＵＳＢ、ＨＡＶｉなどのようなその他の発展段階のネットワーク構想は、単に、ネットワーク接続に関するさまざまなプロトコルを提供しているだけで

ある。これらの通信プロトコルは当業者に周知である。しかし、現在のところ、競合するさまざまなネットワーク技術およびプロトコルの相互運用性の問題点を解決し、ネットワークプラグアンドプレイを達成する統一フレームワークシステムは存在しない。

【００１８】

【発明が解決しようとする課題】したがって、さまざまなタイプのネットワークデバイスを相互接続し、デバイスの発見および利用のための一様な機構を提供する統一フレームワークに対して、強く幅広い受容が認識されており、このようなものがあることは非常に有利である。このような統一フレームワークは、ユーザアプリケーションが、アプリケーション－サービス間通信プロトコル、所望のサービスを提供するデバイスに接続されたホストのＩＰアドレスの発見、およびさらには、ネットワークサービスのユーザインタフェースの詳細のような低レベルの詳細を組み込むことを不要にするとともに、上記のその他の要求を満たす。

【００１９】

【課題を解決するための手段】したがって、本発明の１つの目的は、既知のネットワーキングシステムの上記の制限を克服し、さまざまな機器を相互接続する相異なる通信プロトコルを有する複数のネットワークを単一の統一フレームワークへと統合して、ユーザアプリケーションがさまざまなネットワークデバイスを発見し利用することを可能にする方法および装置を提供することである。

【００２０】具体的には、本発明の目的は、利用可能な機能の変更およびユーザ相互作用に応答してネットワークノードの動的な適応および再構成を達成する方法および装置を提供することである。

【００２１】本発明のもう１つの目的は、アプリケーション、サービス、およびデバイスが、自己の能力および通信インタフェースを記述し、それを他のアプリケーション、サービス、およびデバイスに公表する方法を提供する。

【００２２】本発明のさらにもう１つの目的は、物理的に相異なるデバイスが、接続し、情報交換し、データタイプを交渉し、それぞれの動作についてのステータスを提供して、ネットワークプラグアンドプレイを達成することを可能にする、プラットフォーム独立でトランスポート独立なプロトコルを提供する方法および装置を提供することである。

【００２３】上記の目的を実現し、本発明の効果を達成するため、アクティブコンフィグレーションフレームワークシステムが提供される。本発明のフレームワークは、サービスとサービスユーザを相互接続する。また、このフレームワークは、プラグアンドプレイブローカを有する。本発明のフレームワークでは、サービスユーザは、プラグアンドプレイブローカを用いて、ネットワーキングサービスを発見し、利用し、ネットワーキングサービスと通信する。

【００２４】本発明のフレームワークでは、サービスユーザは、サービスを提供するデバイスによって使用される通信プロトコルとは独立に、プラグアンドプレイブローカのインタフェースを通じてサービスと通信する。

【００２５】また、本発明によれば、サービスをプラグアンドプレイブローカに登録するゲートウェイが提供される。

【００２６】また、本発明によれば、サービスエージェント、ユーザエージェント、およびディレクトリエージェントを有するアクティブコンフィグレーションフレームワークが提供される。本発明のフレームワークでは、サービスエージェントはサービスをディレクトリエージェントに登録する。その後、ユーザエージェントは、要求するサービスの記述をディレクトリエージェントに通信し、ディレクトリエージェントは、適合するサービスのロケーションを帰す。

【００２７】本発明のフレームワークでは、サービスは、デバイスクラスタおよびゲートウェイからなることが可能である。サービスエージェントは、このゲートウェイを通じてデバイスクラスタと通信する。

【００２８】また、本発明によれば、新たなネットワークデバイスの追加に応答して、フレームワークの自動的再構成を行う方法が提供される。本発明の方法によれば、サービスエージェントは、サービスをディレクトリエージェントに登録する。その後、ユーザエージェントは、要求するサービスの記述をディレクトリエージェントに通信する。最後に、ディレクトリエージェントは、要求されたサービスの記述を、提供するサービスの記述と照合し、適合したサービスのアドレスをユーザエージェントに返す。

【００２９】また、本発明によれば、フレームワークに追加される新たなネットワークデバイスによって実行されるサービスに対する通信インタフェースを生成する方法が提供される。本発明の方法によれば、デバイスは、その通信インタフェースを記述するスキーマを生成する。このスキーマは、ＸＭＬ(eXtensible Markup Language)文書の形式とすることが可能である。その場合、ユーザアプリケーションは、ＸＭＬ言語パーサを用いて、通信インタフェースを生成する。

【００３０】また、本発明によれば、ネットワークデバイスを制御する方法が提供される。本発明の方法によれば、ユーザアプリケーションは、デバイス記述スキーマを編集し、編集されたデバイス記述スキーマをデバイスに返す。これに応答して、デバイスは、編集されたデバイス記述スキーマに従って状態を変更する。

【００３１】

【発明の実施の形態】すべてのタイプの分散デバイスおよび分散サービスにわたりネットワークプラグアンドプ

レイを達成するためには、統一フレームワークが必要である。ここで使用される「フレームワーク」あるいは「サブストレート」という用語は、１つ以上のネットワークを統合するシステムを意味する。しかし、多くの場合、「フレームワーク」、「サブストレート」、および「ネットワーク」という用語は区別なく用いられる。

【００３２】本発明のフレームワークを使用することにより、ネットワーク内の相異なる物理デバイスの個数にかかわらず、ネットワーキングエンティティの相互作用のための複雑な機構は一度だけ実装すればよい。さらに、これらの機構は、デバイス固有のゲートウェイを通じて管理ブローカと相互作用するユーザアプリケーションから隠蔽される。本発明のフレームワークは、ＳＬＰ、ＨＴＴＰ、ＸＭＬ、ＪａｖａＲＭＩ、およびＩＩＯＰのような周知のインターネット関連プロトコルに基づいており、ウェブサーバへのアドオンとして実装可能であるため、配備および利用が容易である。さらに、本発明のフレームワークでは、サービス発見の全プロセスは、ユーザアプリケーションにとって透過的（トランスペアレント）であり、物理接続（コネクション）に対する区別がない。例えば、ネットワークに接続されるプラグアンドプレイデバイスはＴＣＰ／ＩＰベースである必要はない。サービス利用プロセスもまたトランスペアレントであり、対象となるさまざまなデバイスの能力と、ユーザの認証および認可に基づく。互換性のないデバイスを接続するときには、トンネリングや完全なデータ変換が必要となることもある。

【００３３】本発明は、デバイス独立な情報交換を容易にするためのアクティブコンフィグレーションフレームワークを提供する。複数のデバイスが接続し、情報交換し、データタイプを交渉し、それぞれの動作についてのステータスを提供するには、相異なるプロトコルが要求される。これらのプロトコルは、プラットフォーム独立であり、その形式や機能にかかわらず、他のデバイスに接続する必要のある任意のデバイスに組み込むことができる。これらのプロトコルはトランスポート独立でもあるため、任意のデバイスが、他の任意のデバイスに接続し、情報交換することが可能となる。

【００３４】本発明のフレームワークでは、アクティブネットワークの場合のように、ユーザ定義計算がネットワーク内に配置される。しかし、アクティブネットワークとは異なり、現在のインターネットアーキテクチャのすべてのルーティングおよびフォワーディングのセマンティクスは、計算環境をアプリケーション層に制限することにより保持される。このフレームワークを利用するネットワークデバイスおよびサービスは、ネットワークインフラストラクチャに深く組み込まれる場合でも、実際には、エンドシステム上のユーザアプリケーションによって作成され、設定され、動的に制御される。このようなアプリケーション定義エンティティは、ネットワーク内の任意のノードで実行され、個々のパケットは、ネットワークを通じて伝搬するとともに任意のアクションを実行するようにプログラムされることが可能である。

【００３５】［アクティブコンフィグレーションフレームワークのコンポーネント］完全なアクティブコンフィグレーションアーキテクチャは、ダミーデバイス、ゲートウェイデバイス、および、ネットワークに直接的または間接的に接続されたＰｎＰブローカからなる。アクティブコンフィグレーションフレームワークのアーキテクチャを図１に示す。図１に示すように、ネットワークには１個以上の実行環境１０２がある。実行環境１０２は、例えば、インターネットのようなネットワーク１０１を用いて相互接続される。実行環境１０２は、ある意味で、ネットワークノードとして作用する。実行環境１０２は、当業者に周知のＪａｖａ仮想マシンやＷｉｎｄｏｗｓ™ベースのマシン内にあることが可能である。実行環境１０２は、ゲートウェイ（図示せず）を通じて非インテリジェント（すなわち、ダミー）デバイスに接続された１個以上のＰｎＰブローカ１０４を有する。ユーザアプリケーション１０３は、ＰｎＰブローカ１０４を通じて、デバイスを発見し、利用し、デバイスと通信する。以下で、本発明のアクティブコンフィグレーションフレームワークシステムのさまざまなコンポーネントについて詳細に説明する。

【００３６】［ダミーデバイス］ダミーデバイスは、ＴＣＰ／ＩＰプロトコルスタック、もしくはＪａｖａ仮想マシン、またはその両方のないデバイスである。このようなデバイスは、ネットワークから直接にアクセス可能ではなく、鋏小物としてゲートウェイを必要とする。このようなデバイスは、ネットワークにインストールされる前に、特定のコンフィグレーション選択を行う必要がある。例えば、それぞれのＸ－１０デバイス（白熱電球など）には、電源線ネットワークに接続する前に手動でハウスコードおよびデバイスコードを割り当てなければならない。さらに、このようなデバイスは、提示するサービスのためのセキュリティやアクセス制御を実施することはない。このようなデバイスのほとんどは、メモリや処理能力を有しておらず、ネットワークの残りの部分に依然として接続する必要のあるレガシーデバイスとして特徴づけることができる。

【００３７】［ゲートウェイデバイス］通常のネットワークデバイスは、単一のネットワークタイプ（例えば、ＩＰまたはＩＰＸ）により、ピアツーピア方式で相互作用することができる。これは、あるネットワークタイプのデバイスは、同じネットワークに物理的に接続された、同じネットワークタイプのデバイスとのみ通信することができることを意味する。第１のデバイスが、その第１のデバイスによってサポートされていないネットワーク上にある第２のデバイス、あるいは、第１のデバイスによってサポートされていないプロトコルを実行して

15

いる第２のデバイスに接続することを必要とする場合に問題が生じる。この問題を解決するには２つのアプローチがある。

【００３８】第１のアプローチは、複数の通信スタックを単一のネットワークデバイスにロードすることである。これにより、デバイスは、適当な通信スタックを有する限り、任意のタイプのネットワークデバイスと相互作用することが可能となる。基礎となるトランスポートは異なるが、デバイスがプロトコルスタック全体を理解する場合には、このアプローチはうまくいく。

【００３９】複数の相異なるデバイスを接続する問題を解決するため、本発明のフレームワークシステムはゲートウェイを使用する。ゲートウェイは、ネットワーク内に不可視的に存在し、あるネットワークタイプを別のネットワークタイプに変換するデバイスである。このアプローチは、デスティネーションデバイスがレガシーであり、各ソースデバイスがデスティネーションと通信するためにデスティネーションをモデル化しなければならない、という本発明の実施例では用いられる。これは、例えば、ファックスデバイスや電子メールデバイスの場合である。デバイスにレガシーネットワークと対話する能力を追加するのではなく、この機能はゲートウェイに置かれる。ゲートウェイは、通常、プログラム可能なプラットフォーム上にホスティングされる。

【００４０】本発明のフレームワークでは、ＰｎＰブローカが、アクセスのためにゲートウェイを要求するレガシー（ダミー）デバイスとのセッションを確立したい場合、ゲートウェイは、ＰｎＰブローカから、そのデバイスの存在を隠蔽する。ＰｎＰブローカは、その代わりに、ゲートウェイとのセッションを作成し、ダミーデバイスのアドレスをゲートウェイに渡す。この時点以降、ゲートウェイは、セッションにおけるいずれかのチャネルを通じてデータを受け取ると、そのデータを、ゲートウェイ－ダミーリンクを通じてリモートデバイスへと中継する。本発明は、次の２つのタイプのゲートウェイを使用する。一般化ゲートウェイは、ＩＰデバイスとの通信を行い、特殊化ゲートウェイは、ダミーを通じて非ＩＰデバイスとの通信を行う。特殊化ゲートウェイは、ダミーデバイスをネットワークに公開するものであり、そのデバイスを制御・管理するためのプログラミングロジックを含む。さらに、特殊化ゲートウェイは、ダミーデバイスのセキュリティポリシーを実現する。１つのゲートウェイデバイスが複数のダミーデバイスを管理することが可能である。

【００４１】［ＰｎＰブローカ］ＰｎＰブローカは、アプリケーション、サービス、およびデバイスのサービスブローカとして作用する。ＰｎＰブローカは、分散システムのためのＩｎｆｏｂｕｓの概念を拡張し、ネットワーク全体をSoftware Busとして扱う。Ｉｎｆｏｂｕｓシステムは当業者に周知である。詳細な説明は、Reaz Hoque, "ConnectingJavaBeans with InfoBus", Wiley Computer Publishing, Nov. 98、にある。Software Busもまた当業者に周知であり、Brian Oki, Manfred Pfluegl, Alex Siegel, and Dale Skeen, "The Information Bus - an architecture for extensible distributed systems", Proceedings of the 14th Symposium on Operating Systems Principles, p.58-68, Asheville, North Carolina, December 1993、に記載されている。

16

【００４２】ＰｎＰブローカにより、ネットワーク接続されたエンティティは、他のネットワーク接続エンティティ（ゲートウェイあるいはダミー）の能力を発見し利用することが可能となる。ゲートウェイは、その能力を、対応するＰｎＰブローカに登録し、ダミーデバイスは、ゲートウェイを発見し、このゲートウェイを通じてＰｎＰブローカと通信する。ネットワーク内のデバイスの各タイプ（例えば、Ｘ－１０Ｘ、１３９４Ｘ、ＵＳＢ）ごとに特定のゲートウェイがあり、その一端でこのデバイスクラスタに接続され、他端でＰｎＰブローカに接続される。Ｘ－１０、ＨＡＶｉ、ＩＥＥＥ－１３９４、およびＵＳＢは周知の通信プロトコルである。２つの相異なるゲートウェイが互いに対話したい場合、通信のためにＰｎＰブローカを使用することができる。例えば、あるゲートウェイがＪｉｎｉクラスタを代表する一方で、別のゲートウェイが、ＨＡＶｉを用いたＩＥＥＥ１３９４クラスタのゲートウェイとなることが可能である。本発明のフレームワークでは、ゲートウェイは、そのアーキテクチャ用に設計されたダミーを使用することにより、受動（非ＴＣＰ／ＩＰ）デバイスと通信する。

【００４３】図２に、アクティブコンフィグレーションフレームワークの全体設計を示す。実行環境１０２内で動作するＰｎＰブローカ１０４は、ゲートウェイ２０１を通じて、ダミーデバイス２０３およびユーザアプリケーション２０２と通信する。ゲートウェイデバイス２０１は、ネットワークノード２０５上で、そのオペレーティングシステム環境内で動作する。ノード２０５は、ゲートウェイおよびサービスへのアクセスを制御するためにセキュリティ実施機構２０４を提供する。

【００４４】ネットワーク接続されたエンティティは、サービスユニットと呼ばれる有意味な機能に再分割される。原理的には、アーキテクチャは、ユーザアプリケーションの観点から、１つの有意味な機能を１つのサービスユニットとして定義する。サービスユニットは、ユーザアプリケーション／サービス全体であることも可能であり、ユーザアプリケーション／サービスの一部であることも可能である。コマンド、応答、およびデータは、ユーザアプリケーションとサービスユニットの間、サービスユニットとサービスの間、あるいはサービスユニットと別のサービスユニットの間で交換することが可能である。サービスユニットは、その機能をＰｎＰブローカに登録する。

【００４５】ＰｎＰブローカにより、ネットワーク接続されたエンティティは、他のネットワーク接続エンティティの能力を発見し利用することが可能となる。ネットワーク接続エンティティは、サービスユーザ（ユーザアプリケーション）であることが可能である。ユーザアプリケーションは、ＰｎＰブローカを通じて、サービスを発見し、それを使用することを要求する。ＰｎＰブローカは、サービスユニットとユーザアプリケーションを接続するためのトランスポート独立なインタフェース（ＰｎＰブローカインタフェースという）を提供する。このインタフェースは、ＴＣＰ／ＩＰプロトコルスタック上で動作するように設計される。ＰｎＰブローカは、サービスブローカまたはマネージャとしてその役割を果たすために、異なるクラスタ内の他のＰｎＰブローカと通信する。ＰｎＰブローカ間のプロトコルはＰｎＰブローカ間プロトコルと呼ばれる。

【００４６】各デバイスクラスタは、高々、１つのＰｎＰブローカを有する。ローカルなＰｎＰブローカがないとき、ユーザアプリケーション／サービスユニットは、ＪａｖａＲＭＩやＨＴＴＰのようなインターネット通信プロトコルを通じてリモートＰｎＰブローカ（別のゲートウェイに接続されたＰｎＰブローカ）を使用することが可能である。ＪａｖａＲＭＩプロトコルは当業者に周知である。その詳細な説明は、Downing, Troy Bryan, "Java RMI: Remote Method Invocation", IDG Books Worldwide, 1998, にある。ＨＴＴＰプロトコルも当業者に周知である。その説明は、"Paul S. Hethmon, "Illustrated Guide to HTTP", Manning Publications, 1997、にある。サービスユニットは、それに直接に接続されたＰｎＰブローカを使用することも可能であるし、ＰｎＰブローカ間プロトコルを使用することによりリモートＰｎＰブローカを使用することも可能である。図３に示すネットワークシステムでは、サービスユニット４０６は、ＰｎＰブローカ間プロトコル４０４を通じて、リモートＰｎＰブローカ１１０４に接続されたサービス１４０６および１４０２を利用する。こうして、あるサービスユニットが、異なるデバイスクラスタに位置するサービスユニットで利用可能なサービスに問い合わせ、これを使用することができる。

【００４７】［ＰｎＰブローカ機能コンポーネント］ＰｎＰブローカの内部コンポーネントを図４に示す。通常、ＰｎＰブローカは、サービスレジストリ３０５、サービスディスカバリ／アベイラビリティエージェント３０４、サービスセッション管理エージェント３０６、サービスロケーションプロトコル（ＳＬＰ：service location protocol）ユーザエージェント３０８およびＳＬＰサービスエージェント３０７からなる。ＰｎＰブローカ間プロトコル３０３は、他のＰｎＰブローカ３０１との通信に使用される。ＰｎＰブローカインタフェース３０２は、ユーザアプリケーション１０３に対するインタフェースを提供する。サービスロケーションプロトコル（ＳＬＰ）は、当業者に周知であり、James Kempf and Pete St. Pierre, "Service Location Protocol for Enterprise Networks: Implementing and Deploying a Dynamic Service Resource Finder", John Wiley and Sons, 1999、に記載されている。サービスレジストリは、１個以上のサービスレコードを含む。サービスレジストリ内のすべての情報は、検索、索引づけおよび交換を容易にするために、ＸＭＬ(eXtensibleMarkup Language)として記述される。ＸＭＬもまた当業者に周知であり、Natanya Pitts-Moultis and Cheryl Kirk, "XML black book", Coriolis Group Books, 1999、に記載されている。

【００４８】［サービスレジストリ］ＰｎＰブローカは、サービスについての情報を保持するためにサービスレジストリを含む。最小限、レジストリは、そのＰｎＰブローカに直接に接続されたゲートウェイおよびサービスについての情報を格納する。図３に示すように、これらのサービスは、ローカルＰｎＰブローカ１０４またはリモートＰｎＰブローカ１１０４にある。さらに、ＰｎＰブローカレジストリは、他のＰｎＰブローカに登録されているサービスについての情報を格納することも可能である。この拡張されたレジストリ機能により、ローカルＰｎＰブローカは、サービスのローカルなディレクトリを保持することが可能となる。これは、ローカルな環境にとって重要であり、ネットワーク内に別個の「集中化した」ＳＬＰディレクトリエージェントは不要となる。例えば、ローカルＰｎＰブローカがプリントサーバをサポートする場合、レジストリは、すべての準拠プリントサーバについての情報を有することが可能である。最終的に、この情報は、負荷均衡、フォールトトレランス、あるいはキャッシュのために使用可能である。また、ＰｎＰブローカサービスレジストリは、デバイスタイプにかかわらず、ローカルＰｎＰブローカの範囲内にあるすべての準拠デバイスについての情報を提供する。ネットワークディレクトリとして作用することも可能である。この場合、ネットワーク内のＰｎＰブローカのうちの１つが、すべてのＰｎＰブローカを見つけて登録する責任を負う中心ディレクトリとして指定されることになる。ネットワークリソースに対する他のデバイスからのすべての要求は、このＰｎＰブローカへ向けて送られ、このＰｎＰブローカはそれぞれに応答することになる。

【００４９】［サービスレコード］サービスレコードは、ネットワーク接続されたクラスタ内の利用可能なサービスユニットのセットであり、利用可能なサービスと、他のＰｎＰブローカから要求されたサービスとを記述する。サービスレコードは、そのローカルクラスタ内の０個以上のサービスユニットレコードからなる。サービスユニットレコードは、サービスユニットＩＤフィー

19

ルドと、それに続く０個以上の属性レコードからなる。サービスユニットＩＤフィールドは、サービスユニットのタイプと、そのサービスユニットによって提供されるサービスとを識別する。デバイスは、ただ１つのサービスユニットを有するが、複数の属性を有することも可能である。属性レコードは、属性ＩＤフィールド、その値、および、アクセス制御情報からなる。属性レコードは、主に、それぞれのサービスまたはデバイスに対する細かいアクセス制御を実施するために使用される。さらに、属性レコードは、デバイスの現在の状態、および、それを変更するために使用可能なインタフェースの記述を格納する。

【００５０】［サービスディスカバリ／アベイラビリティエージェント］ＰｎＰブローカは、リモートＰｎＰブローカを発見し利用可能なサービスを識別するために、サービスディスカバリ／アベイラビリティエージェントを必要とする。サービスディスカバリ（発見）は、ローカルＰｎＰブローカによって指定される、要求されるサービスタイプを、リモートＰｎＰブローカ上で利用可能なサービスタイプと比較することによって実行される。ローカルＰｎＰブローカからリモートＰｎＰブローカへ要求サービスタイプを送信し、リモートＰｎＰブローカからローカルＰｎＰブローカへその応答を送信するために、上記のＪａｖａＲＭＩやＨＴＴＰを使用することができる。要求サービスタイプの指定の検査により、ＰｎＰブローカは、リモートＰｎＰブローカに登録されているすべてのあるいは特定のサービスの特性を決定することができる。サービスディスカバリ／アベイラビリティエージェントは、ＳＬＰユーザエージェントおよびＳＬＰサービスエージェントの上に位置する。

【００５１】さらに、ＰｎＰブローカは、サービスディスカバリ／アベイラビリティエージェントを用いて、サービスのアベイラビリティ（利用可能性）を周期的にチェックすることができる。ローカルＰｎＰブローカは、適当なＰｎＰブローカに対して、アベイラビリティチェックを実行するよう要求する。本発明の実施例では、ユーザは、アベイラビリティチェックの周期を指定し、また、このチェックを取り消すことができる。

【００５２】［サービスセッション管理エージェント］ユーザアプリケーションがサービスまたはデバイスを発見し、それを使用したいとき、ＰｎＰブローカは、ユーザアプリケーションとそのサービスの間に仮想パイプ（トンネル）を確立する。これをサービスセッションという。このようなトンネルを用いて、ユーザアプリケーションとサービスの間で、コマンド、応答およびデータを交換することができる。デバイスのインタフェースの編成に従って、これらのコマンド、応答、およびデータは特定のフォーマットを有し、定義されたプロトコルのもとで交換される。ＰｎＰブローカは、この仮想パイプを管理しながら、以下の３つのプロトコルのうちの１つ

20

で動作するよう指令されることが可能である。

【００５３】［ネイティブパケットでのネイティブデータ転送］ＰｎＰブローカは、データパイプを設定した後、バックグラウンドに入ることにより、ユーザアプリケーションとサービスが、メッセージストリームおよびデータフォーマットを管理することを可能にする。このアプローチは、ＰｎＰブローカが他のネットワークエンティティの能力を発見するために単独で使用され、アプリケーション、サービス、およびデバイスがユーザアプリケーションと発見されたサービスとの間の相互作用を管理するときに有用である。メッセージは、ＰｎＰブローカの関与なしに、ユーザアプリケーションとサービスの間で直接に交換される。メッセージ交換は、厳密に、ネイティブパケットにおけるネイティブデータの形式である。サービスを要求する前にサービスを発見するためあるいはサービスの問合せ（クエリ）を行うために、ユーザアプリケーションは、ＰｎＰブローカ間プロトコルを使用可能である。例えば、ＩＥＥＥ１３９４対応のカメラは、このデータ交換フォーマットを用いて、データストリームをディジタルＶＣＲに送ることができる。

【００５４】［ＰｎＰブローカのパケットにおけるネイティブデータ転送（トンネリング）］データフォーマットはユーザアプリケーションおよびサービスによって選択され制御される一方で、ＰｎＰブローカが、データパイプを設定し、メッセージストリームを管理することも可能である。このアプローチは、ユーザアプリケーションと発見されたサービスとの間で共通のメッセージングプロトコルが存在しないときに有用である。すべてのメッセージは、ＰｎＰブローカ間プロトコルによって運ばれる。例えば、ＩＥＥＥ１３９４対応カメラは、ＰｎＰブローカを通じて、異なるクラスタ内のディジタルＶＣＲにデータストリームを送る。

【００５５】［ＰｎＰブローカのパケットにおけるＰｎＰブローカのデータ（完全機能ゲートウェイ）］ＰｎＰブローカが、データパイプを設定し、メッセージストリームを管理し、ユーザアプリケーションとサービスとの相互作用のためのデータフォーマット定義を提供する。また、ブローカは、ユーザアプリケーションと発見されたサービスとの間の共通のメッセージングプロトコルおよび共通のデータフォーマットを提供する。メッセージ交換情報は、ＰｎＰブローカデータに含まれ、これはパケットにフォーマットされる。ユーザアプリケーションメッセージは、ＰｎＰブローカを通るが、ＰｎＰブローカは、メッセージの内容あるいはセマンティクスを検査することはない。このタイプの相互作用の例は、ＵＳＢデバイスと対話するＩＥＥＥ１３９４対応デバイスである。

【００５６】［ＳＬＰユーザエージェント］ユーザエージェントは、あるサービスとの接続を確立するためにユーザに代わって動作するプロセスである。ユーザエージ

エントは、サービスエージェントまたはディレクトリエージェントからサービス情報を取得する。

【００５７】［ＳＬＰサービスエージェント］サービスエージェントは、サービスを公表するために、１個以上のサービスに代わって動作するプロセスである。

【００５８】［プロトコルと相互作用］ＰｎＰブローカは、ユーザアプリケーションがネットワークのプラグアンドプレイ機能を利用することができるようにする標準化されたインタフェースを公開する。このＰｎＰブローカインタフェースを用いて、アプリケーションは、トランスポート層とは独立にＰｎＰブローカ間プロトコルを用いて相互に通信するように書くことができる。この設計は、ポータビリティ（可搬性）を促進し、ネットワークサービスの発見および選択のためのスケーラブルなフレームワークを提供する。ＰｎＰブローカ間プロトコルは、IETF Service Location Protocol v.2に基づいており、サービスをサポートするネットワークホストの名前をユーザアプリケーションが知ることは不要である。むしろ、ユーザは、ＰｎＰブローカインタフェースを通じて、所望のサービスタイプと、対応する属性のセットとをＰｎＰブローカに与える。これらは、ＰｎＰブローカに対して、所望のサービスを記述する。この記述に基づいて、ＰｎＰブローカは、ＰｎＰブローカ間プロトコルを用いてサービスのネットワークアドレスを解決し、ＰｎＰブローカ間プロトコルは、サービスロケーションプロトコルを使用する。また、ＰｎＰブローカインタフェースは、デバイスあるいはサービスのためのユーザの識別および認証機構を扱う。いったんサービスが識別されると、ＰｎＰブローカ間プロトコルは、ＪａｖａＲＭＩあるいはＨＴＴＰを用いて、ユーザアプリケーションが、発見されたサービスを使用することを可能にする。このフレームワークのもう１つの重要な特徴はゲートウェイによって提供される。ゲートウェイは、デバイスの能力をＰｎＰブローカに登録し、ＰｎＰブローカが、ＰｎＰブローカにおいてサービスエージェントを通じてサービスを使用することを可能にする。

【００５９】要約すれば、本発明のプロトコルスイートは、以下のプロトコルおよびインタフェースからなる。

・ＰｎＰブローカとユーザアプリケーションの間のインタフェース。

・あるＰｎＰブローカと別のＰｎＰブローカの間のプロトコル。

・ＰｎＰブローカとゲートウェイの間のプロトコル。

・ゲートウェイとダミーの間のプロトコル。

【００６０】［ユーザアプリケーションへのＰｎＰブローカのインタフェース（ＰｎＰブローカインタフェース）］ＰｎＰブローカは、サービス登録、サービス発見、サービス要求、およびアベイラビリティチェックのために、ユーザアプリケーションに対して、標準化されたＡＰＩおよびインタフェースを公開する。さらに、ＰｎＰブローカは、適当なサービスあるいはデバイスのためのユーザの識別および認証を扱う。

【００６１】［サービス登録］必要なとき、サービスユニットまたはユーザアプリケーションは、それぞれ、サービスユニットまたはユーザアプリケーションとしてローカルＰｎＰブローカに自分を登録または登録解除するために、RegisterService()またはUnregisterService()を呼び出す。ユーザアプリケーションまたはサービスユニットがＰｎＰブローカに登録された後、その機能は、別のユーザアプリケーションまたはサービスユニットからのQueryService()またはSearchService()要求に対する応答に含まれる。

【００６２】［サービス発見］ユーザアプリケーションは、SearchService()を呼び出して、ローカルＰｎＰブローカに対し、特定のサービスを有する登録されたサービスユニットを含むＰｎＰブローカを探索するよう要求する。ユーザアプリケーションは、QueryService()を呼び出して、どのようなサービスユニットが登録されているか、および、ユーザアプリケーションによって指定されたＰｎＰブローカに登録されているサービスユニットの機能を発見する。

【００６３】［サービス要求］ユーザアプリケーションまたはサービスユニットは、OpenService()を呼び出して、ローカルＰｎＰブローカに対し、ローカルＰｎＰブローカまたはリモートＰｎＰブローカのいずれかに登録されている特定のサービスユニットとのサービスセッションを開始するよう要求する。TransferData()、ReceiveData()、およびCloseService()コールにより、追加機能が提供される。

【００６４】［アベイラビリティチェック］ユーザアプリケーションまたはサービスユニットは、StartAvailabilityCheck()を呼び出して、ローカルＰｎＰブローカに対し、ローカルＰｎＰブローカまたはリモートＰｎＰブローカのいずれかに登録されている特定のサービスユニットが動作しているかどうかを周期的にチェックするよう要求する。

【００６５】［ユーザの識別および認証］好ましい実施例では、本発明は、Ｊａｖａアプリケーション環境を利用する。これは、分散コンピューティングのための適切な、現在利用可能なコンピューティングプラットフォームを提供するからである。この環境では、コードおよびデータの両方が、マシン間で移動可能である。この環境は、別のマシンからダウンロードされたコードが、妥当なレベルの信頼性で動作することを可能にする組込みセキュリティを有する。Ｊａｖａアプリケーション環境における強い型付けにより、仮想マシン上で、オブジェクトのクラスの識別は、そのオブジェクトがそのマシン上で発生したものでないときでも、実行可能である。その結果、ネットワークは、必要に応じて移動可能なオブジェクトの流動的なコンフィグレーションをサポートし、オ

ペレーションを実行するためにネットワークの任意の部分を呼び出すことができる。

【００６６】ホームデバイスへの認証なしのアクセスは、フレームワーク全体のセキュリティにとって重大な結果を生じる可能性がある。リソースへの制御されたアクセスは、安全性とセキュリティの基礎である。従って、本発明の一実施例は、Ｊａｖａ仮想マシンおよびＪａｖａプログラミング環境を使用して、１つのデバイスのセキュリティを保証する。システムは、障害のあるデバイスに対処するように設計することが可能であるが、ネットワークセキュリティは、デバイスセキュリティよりも複雑な問題となる。

【００６７】本発明のもう１つの実施例は、コードを参照する手段として、および、基本的なアクセス制御機構として、コードのＭＤ５チェックサムを使用する。ＭＤ５は、コードのチェックサムを生成する暗号化手続きであり、アクセス制御のために使用される。しかし、ネットワークが大規模になると、セキュリティポリシーを指定レコードに名前付けするためのより柔軟な分散機構が必要である。

【００６８】本発明のセキュリティモデルは、プリンシパルとアクセス制御リストという２つの概念の上に構築される。サービスは、あるエンティティ（プリンシパル）に代わってアクセスされる。プリンシパルは、一般に、システムの特定のユーザに由来する。サービス自体が、そのサービスを実装するオブジェクトの識別に基づいて、他のサービスへのアクセスを要求することが可能である。サービスへのアクセスが許可されるか否かは、そのオブジェクトに対応するアクセス制御リストに依存する。

【００６９】［ＰｎＰブローカ間通信プロトコル］ＰｎＰブローカ間通信プロトコルは、２つの部分に分かれる。第１に、ネットワーク内で新たなデバイスおよびサービスを発見するための、プロトコルおよび機構のセットが使用される。発見（ディスカバリ）手続きは、ユーザアプリケーションがどのようにしてネットワークインフラストラクチャを探索するか、および、ユーザアプリケーションがどのようにして自分自身およびそのサービスを登録することができるかを指定するサービスディスカバリプロトコルの部分に基づく。他方、ルックアップ手続きは、ユーザアプリケーションが、集中化されたレジストリがある場合あるいはない場合に、必要とするサービスを探索するためにどのようにしてレジストリに問合せを行うかを指定するプロトコルに基づく。

【００７０】いったんサービスが探索された後、そのサービスを利用するために他のステップが必要になることがある。本発明の実施例では、ユーザアプリケーションは、サービスの品質およびセキュリティ条件を含めて、サービスへのアクセスの交渉を行う。さらに、サービスへのアクセスの交渉が成功した後、ユーザアプリケーションは、ＪａｖａＲＭＩ、ＨＯＰ、あるいはＨＴＴＰプロトコルを使用して、実際にサービスを利用する。

【００７１】［サービスディスカバリ／登録プロセス］ルックアップおよびディスカバリの両方のために、サービスロケーションプロトコル（ＳＬＰ）が使用される。図５に、サービスディスカバリ手続きを示す。ＳＬＰは、自動的に、しかも事前の設定なしで、ユーザエージェント（ＵＡ：user agent）５０２の要求を、サービスエージェント（ＳＡ：Service Agent）５０４によって提供されるサービス５１２と照合する。ＳＬＰは、ＳＡによるサービスの公表と、ディレクトリエージェント（ＤＡ：Directory Agent）５１１により管理されるＳＬＰディレクトリ５０８へのサービスの編成とを処理する。また、ＳＬＰは、サービスがユーザアプリケーションにサービスの能力および設定条件を通知する手段を提供する。

【００７２】ＰｎＰブローカ５０１内のＵＡ５０２は、ユーザアプリケーション１０３によるサービスおよびリソースの要求を理解する。同じくＰｎＰブローカ５０３（同じＰｎＰブローカでも異なるＰｎＰブローカでもよい）にあるＳＡ５０４は、各ネットワークサービスを代表し、ＵＡ５０２にとって利用可能なサービスを行う。ＳＬＰは、動的にサービス属性を保持するため、ＵＡ５０２は、現在の情報を取得することが可能である。サービスは、アプリケーションによって自動的に探索されることも可能であり、あるいは、サービスブラウザでユーザに提示されることも可能である。ＳＬＰは、既存のアプリケーションをサポートするとともに、新たなサービスの公表および発見を容易に可能にする。

【００７３】本発明によれば、サービスは、ゲートウェイによって記述される。ゲートウェイは、ＳＡ内の属性（そのサービスに対応する）の値を設定する。例えば、プリンタは、ポストスクリプトプリンタとして、青色の紙が利用可能なプリンタとして、あるいは、ユーザのオフィスの同じフロアにあるプリンタとして、記述することができる。ＵＡ５０２は、ＤＡ５１１への要求メッセージＳｒｖＲｅｑ５０５において必要なキーワードおよび属性値を指定することによって適当なプリンタを選択し、応答を待つ。ＤＡ５１１は、適当なサービスのネットワークロケーションを含む応答ＳｒｖＲｐｌｙ５０５により応答する。小規模な施設では、ＤＡがないことも可能であり、その場合、要求メッセージは直接にＳＡに送られることになる。これを、ブロードキャストによるディスカバリ（５０７）という。

【００７４】サービスは、ＤＡ５１１に登録することによって自分自身を公表する。サービスの登録は、サービスを記述するすべてのキーワードと属性値対（属性と値からなる対）とからなるリストを含む。また、登録は、リソースへのリースを含む。これによりリースの満期後、サービス情報はＤＡ５１１によって削除される。リ

ース機構は、サービスが永久に公表され続けているのにサービスハードウェアがもはや利用可能でない状況を避けるために実装される。また、明示的な登録解除も、ＳＡの規則正しいシャットダウン手続きの一部として、ＤＡからサービスの情報を削除することが可能である。また、ＳＡが、現在の属性情報を周期的に登録することにより、ＵＡが、サービスに関連するステータス、負荷、温度、あるいはその他の動的特性を確認することも可能である。

【００７５】本発明のフレームワークでは、サービスは、そのサービスタイプに従って分類される。各サービスタイプは、ＳＡ５０４がサービスを記述するために利用可能にする、利用可能なキーワードおよび属性のセットを定義する。サービスブラウザは、まず、ＵＡ５０２にとって利用可能なサービスタイプを発見した後、そのサービスタイプについてＳＡによって公表されているすべての情報を問い合わせることによって、ＵＡ５０２にとって利用可能なすべてのサービスのすべての特性を決定することができる。

【００７６】［ユーザエージェントとサービスエージェントの相互作用］サービス／デバイス登録プロセスの一部として、まず、サービスユニット５１２のゲートウェイは、サービスあるいはデバイスをＳＡ５０４に登録し、その後、ＳＡ５０４は、ＳｒｖＲｅｇメッセージ５０６をＤＡ５１１に送る。この登録は、サービスの特定のインスタンスに対するサービスＵＲＩと、そのサービスを記述する属性値対とを含む。ＤＡ５１１は、さまざまな目的で、このような登録をキャッシュすることが可能である。登録がキャッシュされた後、ＤＡ５１１は、ＳｒｖＡｃｋメッセージにより応答する。また、サービス登録は、寿命（lifetime）を含む。サービスが利用不能になったがそれ自身を登録解除することができなかった場合、寿命値により、ＤＡ５１１は、キャッシュされた登録を満期にすることができる。この状況は、ＳＡ５０４がＳｒｖＤｅｒｅｇメッセージ５０６を発行することができない場合にしか生じないはずである。正常動作中、ＳＡ５０４は、後続のＳｒｖＲｅｇメッセージで周期的にサービスの登録をリフレッシュする。このような「リフレッシュ」メッセージは、いずれかの値が変化した場合には更新された情報を含むことが可能であるが、完全な属性のセットを含む必要はない。

【００７７】ＰｎＰブローカ５０１のＵＡ５０２は、サービスが要求されるときに、ＤＡ５１１に対して要求を行う。ＵＡ５０２がＤＡ５１１を発見することが可能ないくつかの方法がある。起動時にＤＡ５１１のアドレスをＵＡ５０２に静的に設定することに加えて、ＵＡ５０２は、当業者に周知のＤＨＣＰ（Dynamic Host Configuration Protocol）を用いてＤＡ５１１のアドレスを要求することも可能である。第３のオプションとして、リンク上にマルチキャストを行い、ＤＡの公表を受け取るということがある。これらの３つのオプションは非存するが、ＳＬＰが、手動設定なしで動作可能であることが重要である。この理由で、ＵＡ５０２は、ＤＡ５１１を見つけるためにＳｒｖＲｅｑを使用することが可能である。ＳｒｖＲｅｑは、ＳＬＰに対するＩＡＮＡ割当てマルチキャストアドレスにマルチキャストされる。ＩＡＮＡは、当業者に周知のInternet Assigned Numbering Authority標準である。これはマルチキャスト要求であるため、複数のユニキャスト応答を受け取る可能性がある。その結果として得られるＤＡのリすとは、他のサービス要求のために使用可能である。

【００７８】いったんＵＡ５０２がＤＡ５１１のアドレスを得ると、後続のサービス要求は直接にそのエンティティに対して行うことが可能である。プリンタを例とすると、ＵＡ５０２がプリンタサービスを探索しようとする場合、ＳｒｖＲｅｑが作成される。このメッセージは、サービスタイプ"printer"と、オプションの属性および値のリストとを含む。属性値対は、要求されるプリンタのタイプを記述する。このメッセージは、事前に設定された、または、マルチキャストを通じて発見された、ＤＡ５１１へユニキャストされる。ＤＡ５１１は、ＳｒｖＲｅｑを受信し、キャッシュされた登録に対するルックアップを実行して、要求された属性および値と一致するものを見つけようと試みる。その後、ＳｒｖＲｐｌｙが、要求側のＵＡ５０２へユニキャストされる。この応答は、一致結果に依存して、０個以上のサービスＵＲＩを含む。その後、ユーザアプリケーションは、そのサービスＵＲＩを用いてプリンタを見つける。

【００７９】施設内にＤＡ５１１がない場合、ＰｎＰブローカは、同じくラスタ内のＳＡ５０４を見つけることに制限される。これは、多くの小規模な施設では許容できるかもしれないが、ＰｎＰブローカのスケーラブルなアプリケーションの場合、および、複数のクラスタを有する大規模な施設では、ＤＡ５１１を使用しなければならない。さらに、ＳＬＰ　ＤＡ５１１は、ＬＤＡＰベースのディレクトリ５０９へのゲートウェイであることが可能であるため、ＰｎＰブローカインタフェースは、変換スキーマ５１０を用いて、すべてのプロトコルに対する単一のＡＰＩを提供する。

【００８０】ＵＡ５０２は、サービスハンドルを取得したいとき、サービスタイプと、要求する属性の文字列ベースの開合せとをサービス要求として送信する。サービス応答が返されるとき、これは、サービスを指すＵＲＩを含む。ＵＲＩは、ＳＡ５０４のＩＰアドレスを含むか、さもなければ、ＤＮＳがＩＰアドレスへと解決することができる名前を含む。また、ＵＲＩは、ＵＡ５０２がサービスを使用するために必要とするその他の情報を含む。例えばプリンタの場合、キュー名と、プリントサービスをホスティングしているコンピュータのアドレスとが返される。ＵＲＩは、リソースロケーションを表す

ための従来の標準と同様の自然な方法で、サービスを見つけるために使用される。さらに、文字セット問題は、標準的な方法で対処することができる。

【００８１】［ＰｎＰブローカとユーザエージェントおよびサービスエージェントとの相互作用］ＳＬＰを使用することにより、ＰｎＰブローカは、同じタイプの多くの他の公表されたデバイスから適当なサービスの正確な選択をすることができる。これは、ＵＡ５０２によって要求されたキーワードおよび要求された属性値に一致するサービスのみを要求することによって行われる。このようなキーワードおよび属性値は、ＡＮＤ、ＯＲ演算子、一般的な比較演算子により、また、部分文字列マッチングを使用することにより、ブール式へと結合することができる。

【００８２】ＰｎＰブローカは、ユーザへのコンフィグレーション情報の提供を透過的にし、新たなサービスの設定の作業を容易にする。これらはいずれも、システムアドミニストレータを必要とする。ＳＡ５０４が近隣のＤＡ５１１にサービスを公表するように設定された後、ＵＡ５０２は、さまざまなサービスがオペレーションを開始および停止するとともにネットワーク上で変化する条件に、動的に適応することができる。例えば、ウェブアプリケーションは、現時点でたまたま動作中であるシステムとは独立に、適当なプリンタが利用可能である限りそれを見つけることができる。

【００８３】ゲートウェイは、ＳＡ５０４に新たなデバイスあるいはサービスを追加したいとき、そのサービスの利用可能な属性およびキーワードを供給する。ＳＡ５０４は、プログラム的にＳＬＰに登録を行い、サービスの固有のコンフィグレーション情報に基づいて属性に対する値を割り当てることができる。その後、ＳＬＰは、登録（公表）を処理し、ユーザアプリケーションとサービスの間のコネクションの確立を可能にする。

【００８４】新たなサービスタイプを必要に応じて定義することができる。実験的なサービスタイプを限定された用途で配備し、その後に一般用に公開することも可能である。これは、ＳＬＰのネーミング権限機能を用いて実行される。ほとんどの一般的なサービスタイプは、デフォルトで、ＩＡＮＡ(Internet Assigned NumberingAuthority)により標準化されたサービステンプレート（スキーム）定義を使用する。代替サービスタイプを定義することは、任意の代替ネーミング権限に知られているスキーム定義（通常は、ローカル管理者に知られている名前）を使用するようにディレクトリエージェントを設定するのと同様に容易である。例えば、家庭でセキュリティサービスを提供するためには、"service:secure:Door"というサービスタイプを定義することができる。もちろん、このタイプのサービスハンドルを利用するユーザアプリケーションは、"secure"サービスと通信するために、そのサービスのネットワークプロトコルを使用しなければならず、さらに、この特別のサービスタイプのＵＲＩに含まれる情報を解析することができなければならない。これは、任意のユーザアプリケーション／サーバプロトコルの自然な動作に本来的なことである。

【００８５】［ＳＬＰをＬＤＡＰとともに使用すること］ＬＤＡＰｖ３(Lightweight Directory Access Protocol)は、当業者に周知であり、汎用のディレクトリアクセスプロトコルとして普及しつつある。ＬＤＡＰはその名前にLightweight（軽量）とあるが、ＳＬＰはＬＤＡＰよりもさらに「軽量」である。また、ＳＬＰは、自動ディレクトリ管理を提供するため、および、階層的で自由度の少ない名前空間における不適切なリソース配置を必要としないために、リソース管理においてＬＤＡＰよりすぐれている。ＳＬＰにおいて新たなサービスタイプを追加することはＬＤＡＰの場合よりもずっと容易である。タイプによる問合せは、ＳＬＰの場合のほうがＬＤＡＰの場合よりも効率的である。既存のリソースに対する属性記述は、ＳＬＰでは利用可能であるが、ＬＤＡＰでは不可能である。

【００８６】ＳＬＰでは、事前コンフィグレーションなしでのディスカバリが可能である。これに対して、ＬＤＡＰは、はじめに、ディレクトリのアドレスと、ＬＤＡＰが使用するディレクトリスキームの知識を必要とする。ＳＬＰでは、非標準的な属性と、標準化されたサービスタイプテンプレートの新しいバージョンが可能であるため、進化することができる。

【００８７】本発明の実施例は、ＳＬＰを使用して、ＬＤＡＰディレクトリの管理を簡単化し、ＳＬＰが、必要に応じてＬＤＡＰディレクトリから取得される情報を返すことを可能にする。別の実施例では、ＳＬＰは、ＬＤＡＰにサービスエントリを入力するために動的に使用され、ＳＬＰ問合せはＬＤＡＰ問合せにマッピングされて、ＬＤＡＰがＳＬＰのバックエンドとなる。このようなコンフィグレーションの１つの利点は、ユーザアプリケーションが、ＬＤＡＰディレクトリから直接にユーザ認証情報を得ることである。

【００８８】［サービス利用プロセス］サービスディスカバリ／登録プロセスは、サービスロケーションプロトコルによって提供されるＵＲＩによって検索可能なサービスレコードを提供する。ＰｎＰブローカ間の交換能力は、サービスレコードを含むQueryService()メッセージを交換することにより提供される。この問合せ(query)は、ユーザアプリケーションが利用したいサービスの要件を記述し、リモートＰｎＰブローカに対してサービスの詳細を要求する。

【００８９】受信側ＰｎＰブローカは、QueryService()メッセージを受信すると、受信したサービスレコードを自己のサービスレコードと比較する。その後、受信側ＰｎＰブローカは、一致したサービスレコードを含むサービスレコードを要求側ＰｎＰブローカに返す。このアプ

29

ローチは、すべてのサービスユニット、特定のタイプのサービスユニット、あるいは、特定の能力のセットを有するサービスユニットのうちで、ユーザアプリケーションが探索を行う技術を提供する。ユーザアプリケーションは、サービスレコードを受け取った後、サービスあるいはデバイスを使用するためにＰｎＰブローカとの通信を開始することができる。

【００９０】ローカルＰｎＰブローカは、リモートＰｎＰブローカへメッセージを送ることによって、ユーザアプリケーションとサービスの間のサービスセッションの確立を要求する。このメッセージは、ユーザアプリケーションサービスハンドル、リモートサービスユニットハンドル、プロトコルＩＤ（ネイティブ／トンネル／ゲートウェイ経由）、要求側（リクエスタ）ＩＤ、および要求側資格証明(credential)からなる。

【００９１】リモートＰｎＰブローカは、要求されたサービスが受容されるかそれとも拒否されるかを指定するリザルトコードで応答する。リモートＰｎＰブローカは、ある時間後のリダイレクトまたはリトライを含むことも可能である。サービスが受容される場合、リモートＰｎＰブローカは、サービスセッションの開始を識別するサービスハンドルを送信する。サービスセッションは、ユーザアプリケーションとサービスユニットの間、または、２つのサービスユニットの間でのデータ転送を扱う。ユーザアプリケーションは、サービスの使用を完了すると、ローカルＰｎＰブローカに対して、"close service"（サービス終了）メッセージをリモートＰｎＰブローカへ送るよう要求する。さらに、ユーザアプリケーションは、サービスが利用可能であるかどうかを周期的に判定する必要があるとき、それぞれのＰｎＰブローカに対して、フレームワーク内のＰｎＰブローカ間でアベイラビリティチェックを実行するよう要求することができる。アベイラビリティチェックは、１つのデバイス全体に対しても、あるいは、デバイス内で提供される特定のサービスに対しても、実行可能である。

【００９２】［ＰｎＰブローカ／ゲートウェイプロトコル］ＰｎＰブローカ／ゲートウェイプロトコルは、簡単なメッセージ交換機構に基づいている。ゲートウェイは、既に説明したように、ゲートウェイに直接に接続されたサービスユニットを代表し、提供されるサービスを登録する。また、ゲートウェイは、ＰｎＰブローカにあるサービスエージェントに対する、任意のサービスのコンフィグレーションにおいてなされる変更を反映する。この登録は、リースに基づいており、ゲートウェイまたはサービスエージェントのいずれかにより更新可能である。登録は、サービスエージェントに現在の情報を提供する。ＰｎＰブローカは、特定のサービスに対する要求を受け取ると、まず、サービスエージェントをチェックして、必要なデバイスあるいはサービスをいずれかの直接接続されたゲートウェイが提供するかどうかを判定する。また、ゲートウェイにより、サービスエージェントは、ゲートウェイ内の特定のイベントに関心があることを登録し、そのイベントの発生の通知を受け取ることが可能である。このようなイベントは、ネットワークにおけるデバイス／サービスの追加／削除であることが可能である。サービスのサービスレコードがアプリケーションの要件に一致した場合、ユーザアプリケーションは、ゲートウェイ内のサービスユニットと、ユーザアプリケーションとの間のサービスセッションを設定することができる。

30

【００９３】［イベント通知機構］本発明のフレームワークでは、ユーザアプリケーションは、イベント通知要求をサーバに登録することができるため、サーバは、新たなデバイスがオンラインになったとき、あるいは、デバイスの属性が変化したときに、ユーザアプリケーションに通知することになる。これを達成するため、本発明は、登録ユーザにイベント通知を配信するための機構を提供する。

【００９４】ネットワークイベント通知プロトコルには、ＣＯＲＢＡ(Common Request Broker Architecture)イベントサービス、Ｘ－Ｗｉｎｄｏｗシステムイベント、ＳＧＡＰ、ＢＳＣＷなどのいくつかの周知のものがある。例えば、ＣＯＲＢＡイベントサービスは、Robert Orfali and Dan Harkey, "Client/Server Programming with Java and CORBA", Wiley, 1996, に詳細に記載されている。

【００９５】しかし、上記のイベント通知サービスは、特定のアーキテクチャで動作するように設計され、大規模なコードベースを要するため、軽量通知サービスにおいて実際に使用するのは困難である。

【００９６】いくつかのタイプのイベントの通知のための登録に関して提供される能力に加えて、ユーザは、属性を通知要求に関連づけることができる。好ましくは、このような通知は、高い信頼性で配信され、完全に規則正しいエンドツーエンド配信を保証するべきである。しかし、通知が、信頼できないトランスポートを通じて配信されるとき、確認応答やリトライは不要である。ユーザアプリケーションは、イベントの需要側が、受信の明示的な確認を供給側に提供することを要求することも可能である。認証のために、ユーザは、通知にディジタル証明して、通知の正当性および完全性を保証することができる。

【００９７】本発明のフレームワークは、ＳＬＰの使用により、２つの方法でイベント通知を達成する。第１実施例によれば、サービスエージェントは、サービスが更新された後、マルチキャストアナウンス（ＳｒＶＲｅｇ）を送信する。更新を受信した後、ユーザエージェントは、ブラウザ更新を行うことが可能であり、ＤＡは登録を更新することができる。このような使用法は、ＳＬＰ使用では明示的に禁じられていないが、フラッディン

グ(flooding)を引き起こす可能性がある。さらに、このアプローチは、多数のサービス、異なる言語、およびディジタル署名に関して効率的に動作しない。また、ＳｒｖＲｅｇは、ネットワークにおいて追加／削除／変更された各サービスのＵＲＩおよび属性を含む。

【００９８】本発明のもう１つの実施例では、サービスエージェントは、ネットワーク内のすべてのユーザエージェントへのブロードキャストにより、ＳＡＡｄｖｅｒｔｓメッセージをアナウンスする。ＳＡＡｄｖｅｒｔｓメッセージは、提供されるすべてのサービスタイプのリストを含む。ＳＡＡｄｖｅｒｔｓを得た後、ユーザエージェントは、送信側サービスエージェントへ適当なサービス要求をユニキャストする。サービスエージェントからのアナウンスは、指数バックオフ方式で行われる。

【００９９】［ゲートウェイ／ダミープロトコル］ゲートウェイ／ダミープロトコルは、特殊なプロトコルであり、ネットワークにより統合されるデバイスのタイプを反映する。ネットワークは、Ｘ－１０、ＵＳＢおよびＩＥＥＥ１３９４のデバイス／サービスをそれぞれネットワークの残りの部分に接続するＸ－１０ゲートウェイ、ＵＳＢゲートウェイ、およびＩＥＥＥ１３９４ゲートウェイからなることが可能である。ダミーデバイスは、サービスレコードに対応するタイプのレコードを独自(proprietary)フォーマットで格納する。ゲートウェイ／ダミープロトコルは独自のものであるが、ＰｎＰブローカに対しては、標準化されたインタフェースを公開する。

【０１００】［実施例］本発明の一実施例によれば、ネットワークは、１個以上のＰｎＰブローカを有する。ＰｎＰブローカは、ユーザといくつかのサービスとの間のインタフェースとして作用する。一実施例では、Ｘ－１０デバイスに対するＰｎＰブローカは、ＵＳＢデバイスによって提供されるサービスにリンクされたいくつかのボタンあるいはその他のユーザインタフェースウィジェットを提供する。ここで用いた「ユーザインタフェースウィジェット」という用語の意味には、ユーザアプリケーションのグラフィカルユーザインタフェースを作成するために使用されるボタン、ダイアログ、テキストウィンドウ、スケールおよびその他のグラフィカルコンポーネントを含む。あるボタンがクリックされると、ＰｎＰブローカは、単に、Ｘ－１０ゲートウェイを通じて、特定のサービスを呼び出す。以下のステップは、この実施例により、ユーザアプリケーションが、ネットワーク内の新たに登録されたサービスを利用するために必要とされる。

【０１０１】１．白熱電球が、家庭の電源アウトレットに挿入される。

【０１０２】２．白熱電球は、ダミーデバイスであるため、Ｘ－１０デバイスを通じて電源線ネットワークに接続される。

【０１０３】３．ホームネットワークでは、ＰｎＰブローカおよびＸ－１０ゲートウェイは、新たなデバイスの導入前に既に動作中である。

【０１０４】４．Ｘ－１０ゲートウェイは、利用可能なアドレスを周期的にポーリングし、いずれかのデバイスがアクティブになったかどうかをチェックする。

【０１０５】５．ゲートウェイは、新たなデバイスがネットワークに入ったことを認識し、そのデバイスと、そのデバイスにより提供されるサービスとの両方を、ＰｎＰブローカのサービスエージェントに登録する。

【０１０６】６．サービスエージェントは、デバイスを使用するために、あるリースを取得する。このリースは、各リース期間後に更新して、これにより、ゲートウェイに対して、サービスが依然として利用可能であるかどうかをチェックするよう要求しなければならない。

【０１０７】７．サービスエージェントは、デバイスにより提供されるサービスを、サービスロケーションプロトコルを用いて、ディレクトリエージェントに登録する。

【０１０８】８．ユーザは、デバイスを使用したい場合、ＰｎＰブローカインタフェースを用いて、要求するサービスに対する問合せおよび検索を行う。すると、ＰｎＰブローカにあるユーザエージェントは、ディレクトリエージェントと連絡をとるか、あるいは、（サービスエージェントおよびユーザエージェントがいずれも１つのＰｎＰブローカ内にある場合には）直接にサービスエージェントへ行く。

【０１０９】９．新たに発見されたデバイスが、ユーザのデバイス記述と一致する場合、サービスエージェントあるいはディレクトリエージェントは、サービスパラメータとともに、固有のＵＲＩをユーザエージェントに返す。

【０１１０】１０．ユーザアプリケーションは、これが非ＴＣＰ／ＩＰデバイスであると判定し、これに従い、ＰｎＰブローカ内のサービスセッション管理エージェントは、ユーザアプリケーションとデバイスの間のセッションを確立する。しかし、このセッションでは、コマンド／データ転送は、ＰｎＰブローカをゲートウェイとして使用して、ＰｎＰブローカ間プロトコルにより行われる。

【０１１１】１１．また、ユーザアプリケーションにより提供されるインタフェースは、デバイス内の機能の実行に関する情報も含む。

【０１１２】１２．ゲートウェイは、サービスにおける状態の変化をサービスエージェントに知らせて更新する。

【０１１３】図６において、本発明の一実施例による例示的なホームネットワークは、電源線（Ｘ－１０）クラスタ６１３、電話線（ＨｏｍｅＰＮＡ／ｘＤＳＬ）クラスタ６０２および６０６、ＣＡＴ５（イーサネット（登録商標））クラスタ６１６、ＵＳＢクラスタ６１０、な

らびにＪｉｎｉデバイスクラスタ６０９などの、周知のネットワークコンポーネントにより構成される。レジデンシャル（住宅）ゲートウェイ６０５は、イーサネット、ＨｏｍｅＰＮＡ、ｘＤＳＬ、およびＪｉｎｉを含むさまざまなネットワークを相互接続するために使用される。イーサネットクラスタは、ユーザのホームＰＣ６２０を含む。ホームＰＣ６２０は、イーサネットプロトコルにより、ディジタルカメラ６１７およびウェブパネル６１８のようなさまざまなデバイスに接続される。ウェブパネル６１８は、ＮＥＣ社の製品であり、タッチパッドを有し、ユーザに、インターネットに容易にアクセスする手段を提供するデバイスである。イーサネットクラスタ６１６は、イーサネットハブ６１９を通じてレジデンシャルゲートウェイ６０５と通信する。イーサネットハブ６１９もまた、インターネットへのホームネットワークのブリッジを提供する。また、ホームＰＣ６２０は、ＵＳＢ配線を有するスピーカ６１１およびネットワークカメラ６１２に、および、Ｘ－１０配線を用いて白熱電球６１４およびセキュリティマネージャ６１５に接続される。さらに、ホームネットワークは、別のホームＰＣ６０３および６０７、テレビジョン６０１、ならびに電話機６０４を有する。ホームＰＣ６０７を含むｘＤＳＬクラスタ６０６は、ブリッジ６０８を通じてレジデンシャルゲートウェイ６０５に接続される。ＰＣ６２０上で動作するＰｎＰブローカは、ユーザが、ウェブブラウザに基づくインタフェース内で、新たなデバイス／サービスを検索することを可能にする。本実施例は、ユーザインタフェースとして、Microsoft Internet Explorer™ v5.0ウェブブラウザを使用するが、他の適当なブラウザも使用可能である。

【０１１４】デバイス／サービスは、対応するネットワークに挿入されると、設定可能なプロパティとともにブラウザに現れる。デバイス固有のゲートウェイは、デバイス／サービス記述を格納する。ディレクトリエージェントがネットワーク内にある場合、ディレクトリエージェントが、デバイス／サービスの機能の登録を扱う。正当な権限が与えられた後、ユーザは、デバイスのプロパティの変更や、デバイスの相互接続が可能である。ＰｎＰブローカは、デバイスに対するユーザの認証と、非互換の場合のデータ／メッセージの変換を扱う。例えば、Ｘ－１０デバイス６１４および６１５は、ＰＣ６２０上にあるゲートウェイデバイスを使用することによって、フレームワーク内の残りのデバイスと対話することができるダミーデバイスである。ＰｎＰブローカは、ＨＴＴＰサーバへのアドオンとして実装されることも可能である。ユーザは、電源線デバイス６１４および６１５の状態（電灯のオン／オフ）を変更することができる。デバイス（例えば、カメラ６１７）がＴＣＰ／ＩＰベースである場合、ユーザは、カメラにより提供されるインタフェースに基づいて手動設定を実行し、カメラにより撮られる画像を見ることができる。ユーザは、ビデオオンデマンドサーバに対して、利用可能な映画のリストを問い合わせることができる。通常、（ＰＣ６０３上のＭＰＥＧ－２デコーダを用いて）ネットワークに接続されたアナログＴＶ６０１は、同じタイプのインタフェースを使用している場合には、ビデオオンデマンドサーバにも接続可能である。しかし、ＴＶ６０１およびネットワークカメラ６１２はインタフェースを共有していないため、ユーザは、カメラの画像を直接にＴＶで見ることはできない。Ｊｉｎｉデバイス（例えば、プリンタ）がネットワークに接続されている場合、このデバイスは、非ＪｉｎｉデバイスがあたかもＪｉｎｉデバイスであるかのようにして、非Ｊｉｎｉデバイスとともに動作する。

【０１１５】［Ｊｉｎｉインタフェースの問題点に対する解決法］ネットワークプラグアンドプレイを達成する候補のうちの１つとしてのＪｉｎｉインタフェースモデルの欠点については既に説明した。本発明は、Ｊｉｎｉセキュリティモデルを借りながら、上記のＪｉｎｉインタフェースの欠点を克服する。Ｊｉｎｉのインタフェースの問題点を解決するには、以下の４つのアプローチを使用可能である。第１に、標準に基づく相互運用性が提供可能である。このアプローチによれば、すべてのサービスは標準ＡＰＩを有し、サービスは、それらの標準ＡＰＩを実装することになる。第２のオプションは、サンドボックスに基づく相互運用性である。この場合、適当なセキュリティモデルを有するＪａｖａ仮想マシンが与えられれば、サービスは独立して動作することができる。第３のオプションは、リフレクションに基づく相互運用性である。この場合、アプリケーションは、サービスに対して、そのインタフェースについて質問し、リフレクション機構を通じて、このインタフェースの状態に影響を及ぼす。最後に、第４の方法は、実装に基づく相互運用性である。この場合、同じ人あるいは会社が、プロキシおよびサービスの両方を製造することになる。

【０１１６】Ｊｉｎｉとは異なり、コードをクライアントに移動しそれをＪａｖａ仮想機境内で実行する代わりに、本発明は、サービスのアドレスおよびそのサービスのインタフェースを記述するその他のパラメータを処理し、このインタフェース情報をクライアントに提供する。

【０１１７】本発明のフレームワークが提供する主な利点の１つは、利用可能な機能およびユーザ相互作用モードにおける変更に応じたエンドユーザアプリケーションの動的な適応および再構成である。一部のインタフェースモデルは、ユーザが「従来型」ソフトウェアとのみ相互作用することができることに基づいている。このようなシステムでは、ユーザは、アプリケーションをカスタマイズする能力において制約があるため、アプリケーションは、デバイスの能力を最大限に、すなわち、サービス設計者が予測したユーザの需要あるいは提供したプロ

グラムインタフェースの限界まで、利用することができない。

【０１１８】多くの場合、自由度はユーザインタフェースによって制限される。アプリケーションの状態およびアプリケーションの初期設定は、ユーザインタフェースを通じてのみ操作可能であるが、ユーザインタフェース自体はあまり設定の自由度はない。これは、モノリシックなプログラムにおいて顕著であり、アプリケーションの状態にフックを設けたり、適切に定義された外部プロトコルを通じてアプリケーションの状態にアクセス可能にしたりするのがきわめて不便である。これに対して、クライアント／サーバプログラムは、公開された相互作用プロトコルを提供するが、２つの大きな問題点を有する。インタフェースの仕様がしばしば臨時的(ad hoc)であること、および、ネットワーク機能についてのアプリケーションのビューを（クライアントコードを修正することによって）変更することができるのはプログラマのみであることである。

【０１１９】上記のインタフェースの問題点を解決する１つのアプローチは、アプリケーションプログラムが、オンザフライでクライアントデバイスあるいはコンピュータに、例えば「ｊａｖａアプレットとして、ダウンロードされるようにすることである。しかし、このアプローチの問題点として、エンドユーザが、関連するエンティティとして異種のサービスのセットとの相互作用のためにアプリケーションをカスタマイズすることができないことがある。すなわち、このアプローチでは、アプリケーションが透過的でないために、機能的に同一のサービスの場合でも、プロトコルにおける小さい相違を克服することができない。例えば、上記のインタフェースモデルに基づく電灯スイッチ制御プログラムは、新たな電灯スイッチに遭遇するたびに、それを使用するためには異なるアプリケーションをダウンロードする必要がある。したがって、このアプローチは機能を公開するが、操作しやすい形式ではない。

【０１２０】もう１つのアプローチは、さまざまなサービスの機能インタフェースを標準化し、アプリケーションがこれらのインタフェース標準をサポートすることを要求することにより、上記の問題点を回避することである。このアプローチの問題点は、多数のアプリケーション固有の標準を強制することが非実際的である点である。

【０１２１】明らかに、上記のいずれとも異なるモデル、すなわち、インタフェースを公開する要求と、プロトコル標準について合意する要求とのバランスのとれたモデルが好ましい。この問題点を解決するための第１の困難は、サービスの利用可能な機能（インタフェース）を記述する単一ドキュメントスキーマを定義し、関連するプログラムおよびユーザインタフェースをサービスの集合に（およびその逆）関連づけることである。第２の困難は、カスタムユーザインタフェースが利用可能でないときに、上記のスキーマで書かれたドキュメントを用いてユーザインタフェースを生成し、ランタイム環境を実装することができるソフトウェアを提供することである。

【０１２２】本発明の一実施例は、コンポーネントベースのアプリケーションフレームワークを、コンポーネント記述のためのアーキテクチャ独立なドキュメントモデルとともに利用する。このようなコンポーネントベースのアプリケーションフレームワークの詳細な説明は、David Krieger and Richard Adler, "The emergence ofdistributed component platforms", IEEE Computer Magazine, p.43-53, March 1998、にある。このフレームワークは、上記の２つの基本的なアプローチの特徴を組み合わせることにより、コードフラグメントのアップロード／ダウンロードを可能にし、アプリケーション固有でないインタフェースの記述および操作のために標準を課する。

【０１２３】［本発明のアクティブコンフィグレーションモデル］本発明のアクティブコンフィグレーションモデルでは、ＸＭＬ(eXtensible Markup Language)記述が、すべてのネットワークデバイスに関連づけられる。ＸＭＬが使用されるのは、ＸＭＬ記述が、（サーバ側での）デバイスの機能の公表として、静的で不変のインタフェース記述を提供するからである。さらに、このようなＸＭＬサービスインタフェース記述を操作することによって、クライアントは、フレームワーク内のサービスを利用することができる。フレームワーク内、操作のための標準的なセッションを提供するために、それぞれのサービスオブジェクトにプログラムおよびユーザインタフェースを公開する。

【０１２４】図７に、本発明のアクティブコンフィグレーションモデルを示す。本発明によれば、ネットワーク内のあらゆるデバイスあるいはサービス７０１は、その機能インタフェースの定義７０２を指定する。これらのインタフェース定義は、アナウンス７０５によってクライアントに伝えられる。これらのインタフェース定義は、サーバ側で静的（スタティック）である（ＣＯＲＢＡにおけるＩＤＬと同様）。これらのインタフェースは、ネットワーク内のすべてのエンティティ間で共有されるが、いずれのユーザアプリケーションによって操作されることも可能である。ユーザアプリケーションは、サービスインタフェースの使用、および、このようなインタフェースによって提示されるメタデータに対する完全なコントロールを有する。ユーザアプリケーション７０３によってインタフェースまたはその使用の状態に変更７０４があれば、リファレンス（参照）７０６によりデバイスあるいはサービスに反映される。

【０１２５】したがって、本発明は、任意のネットワークサービスを構築するためのプログラム可能な基盤を提

供する。本発明の一実施例では、エンドユーザアプリケーションは、利用可能な機能および相互作用モードにおける変更に応答して動的に適応され再構成される。この実施例は、コードをダウンロードするＪａｖａアプレットの考え方と、標準化されたインタフェースの記述および操作との利点を組み合わせている。本発明のフレームワークでは、ユーザ（あるいはマシンにより生成された ユーザアプリケーション）は、単に、サービスによって提供されるインタフェース記述を編集し、その編集をデバイスに反映させることによって、ネットワークあるいはデバイスの状態を変えることができる。さらに、デバイスあるいはサービスは、標準化されたＡＰＩを通じてアクセス可能である。すべてのＡＰＩを標準化しようとすることによって特徴づけられる従来のシステムとは異なり、本発明のフレームワークによれば、さまざまなネットワーキングデバイスのベンダは、自己の製品の記述をフレームワークにマッピングすることが可能である。こうして、ベンダは、構文的定義およびデバイスの能力に集中することができる。

【０１２６】本発明によれば、デバイス記述は、宣言的スタイルのＸＭＬを用いて格納され、アプリケーションコードに追加して使用される。ＸＭＬデバイス記述の主な機能は、データおよび制御フロー情報を提供することである。この制御／データの分離を公開することにより、本発明は、メタデータを設計に明瞭に組み込むので、格納および操作がアプリケーションとは独立となるため、アプリケーションは、将来の変更が可能なように設計することができる。

【０１２７】本発明のフレームワークで用いられているような、プログラム／ユーザインタフェースを、それらが参照するオブジェクトから分離することは、当業者に周知のＳｍａｌｌｔａｌｋ™のモデル／ビュー／コントローラ（Ｍ／Ｖ／Ｃ）アーキテクチャに類似している。モデル／ビュー／コントローラアーキテクチャについてはさらに詳細には、G. Krasner and S. T. Pope, "A Cookbook for Using the Model View Controller User Interface Paradigm in Smalltalk-80", Journal of Object Oriented Programming, August/September 1988、に記載されている。

【０１２８】Ｍ／Ｖ／Ｃアーキテクチャでは、データ（モデル）は、データの表示（ビュー）およびデータを操作するイベント（コントローラ）から分離される。同様に、本発明のシステムにおける文書は、データを操作し見るためのユーザインタフェース／プログラムにそのデータを関連づける糊（グルー）として作用する。

【０１２９】本発明は、ＸＭＬ構文を利用して、デバイス記述スキーマを、ＸＭＬ文書型定義（ＤＴＤ）として作成する。ＸＭＬは、構造化された情報を含む文書のためのマークアップ言語である。これは、ＳＧＭＬのサブセットであり、階層的な名前付けされた値と、他の文書を参照するための高度な手段の形で、自己記述型のカスタムマークアップを提供する。ＸＳＬ(Extensible Style Sheets)やＸＬｉｎｋ(Extensible Linking Language)のような同類のプロトコルとともに、ＸＭＬは、付随する文書スキーマ（文書型定義：ＤＴＤ）のグループを指定し、発見し、結合する能力を提供する。しかし、ＨＴＭＬとは異なり、ＸＭＬにおけるタグのセットはフレキシブルであり、タグのセマンティクスは、その文書に付随するＤＴＤによって定義される。Resource Description Format、Dublin CoreおよびＸＭＬ－Ｄａｔａのような他のメタデータマークアップ言語も提案されている。

【０１３０】本発明のフレームワーク内では、デバイスあるいはサービスは、自己の記述スキーマを、ＸＭＬ文書と、それに伴うＤＴＤおよびスタイルシートとして作成する。このスキーマは、言語独立なサービス記述のため、ならびに、ユーザインタフェース（プログラム）をサービスオブジェクトに、およびその逆にマッピングするための、マークアップタグを提供する。また、スキーマは、サービスのインタフェースをＸＭＬ／ＸＳＬ定義内に組み込むので、パーサは、ユーザアプリケーションにコードをダウンロードすることを必要とせずに、これらのインタフェースを読み出すことができる。例えば、電灯スイッチのための＜ＯＮ＞タグは、デバイスがメソッド呼出しおよびその他のイベントをリスンするアドレスおよびポート番号を示すことが可能である。

【０１３１】ユーザインタフェースをこれらのサービス記述から生成するために、ＸＭＬパーサが、クライアントユーザアプリケーションによって使用される。クライアントは、簡素的な型をユーザインタフェースウィジェットにマッピングし、ＸＭＬ／ＸＳＬを解析した後、ネットワークの現在のコンフィグレーションに対するユーザインタフェースを生成する。ユーザ（あるいはユーザアプリケーション）は、そのデバイスに対応する動的に生成されるＵＩウィジェットを単にクリックすることによって、任意のネットワークデバイスの状態を変更することができる。このアクションは、ユーザのマシン上で現在のＵＩ状態を変更するとともに、適当なコマンド（デバイスベンダにより定義されＸＭＬ文書に埋め込まれる）を実行のためにデバイスに送る。本発明の実施例では、この目的のために、Ｊａｖａ™で書かれたパブリックドメインのＸＭＬパーサと、Internet Explorer™ 5.0のようなＸＭＬ対応ウェブブラウザが使用される。

【０１３２】以上、本発明について、その好ましい実施例を用いて説明したが、当業者には容易に認識されるように、本発明の技術的範囲および技術思想から離れることなく、形式および細部におけるさまざまな変更が可能である。

【図面の簡単な説明】

【図１】アクティブコンフィグレーションフレームワークのインフラストラクチャの図である。

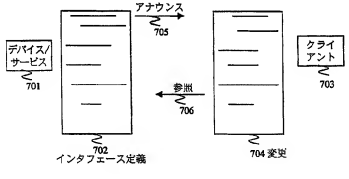
39

【図２】アクティブコンフィグレーションフレームワークの全体設計図である。
【図３】２つのプラグアンドプレイブローカ間の通信を示す図である。
【図４】プラグアンドプレイブローカの内部表現を示す図である。
【図５】サービス発見（ディスカバリ）手続きを示す図である。
【図６】アクティブコンフィグレーションフレームワークの実装例の図である。
【図７】アクティブコンフィグレーションインタフェースモデルの図である。
【符号の説明】
１０１　ネットワーク
１０２　実行環境
１０３　ユーザアプリケーション
１０４　ＰｎＰブローカ
２０１　ゲートウェイデバイス
２０２　ユーザアプリケーション
２０３　ダミーデバイス
２０４　セキュリティ実施機構
２０５　ネットワークノード
３０１　ＰｎＰブローカ
３０２　ＰｎＰブローカインタフェース
３０３　ＰｎＰブローカ間プロトコル
３０４　サービスディスカバリ／アベイラビリティエージェント
３０５　サービスレジストリ
３０６　サービスセッション管理エージェント
３０７　サービスロケーションプロトコル（ＳＬＰ）サービスエージェント
３０８　ＳＬＰユーザエージェント
４０４　ＰｎＰブローカ間プロトコル
４０６　サービスユニット
５０１　ＰｎＰブローカ
５０２　ユーザエージェント（ＵＡ）

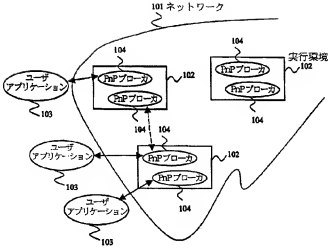
40

５０３　ＰｎＰブローカ
５０４　サービスエージェント（ＳＡ）
５０７　ブロードキャストによるディスカバリ
５０８　ＳＬＰディレクトリ
５０９　ＬＤＡＰディレクトリ
５１０　変換スキーマ
５１１　ディレクトリエージェント（ＤＡ）
５１２　サービス
６０１　テレビジョン
６０２　ＨｏｍｅＰＮＡクラスタ
６０３　ホームＰＣ
６０４　電話機
６０５　レジデンシャルゲートウェイ
６０６　ｘＤＳＬクラスタ
６０７　ホームＰＣ
６０８　ブリッジ
６０９　Ｊｉｎｉクラスタ
６１０　ＵＳＢクラスタ
６１１　スピーカ
６１２　ネットワークカメラ
６１３　Ｘ－１０クラスタ
６１４　自熱電球
６１５　セキュリティマネージャ
６１６　イーサネットクラスタ
６１７　ディジタルカメラ
６１８　ウェブパネル
６１９　イーサネットハブ
６２０　ホームＰＣ
７０１　デバイス／サービス
７０２　インタフェース定義
７０３　ユーザアプリケーション
７０４　変更
７０５　アナウンス
７０６　参照
１１０４　リモートＰｎＰブローカ

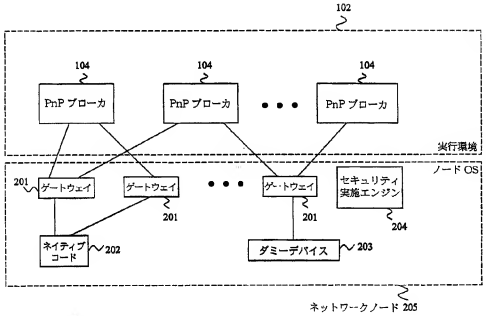
【図７】



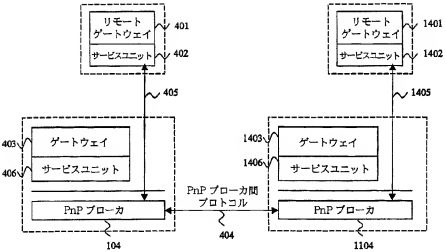
【図１】


101 ネットワーク
104
PnPブローカ
102
実行環境
ユーザアプリケーション
103
PnPブローカ
PnPブローカ
PnPブローカ
PnPブローカ
ユーザアプリケーション
ユーザアプリケーション

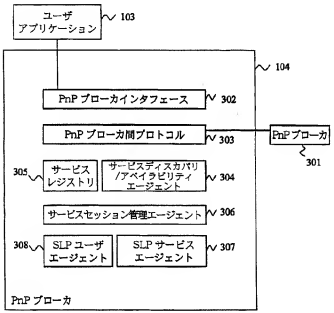
【図２】


102
104
PnP ブローカ
PnP ブローカ
PnP ブローカ
実行環境
ノード OS
201
ゲートウェイ
ゲートウェイ
ゲートウェイ
セキュリティ実施エンジン
204
ネイティブコード
202
ダミーデバイス
203
ネットワークノード 205

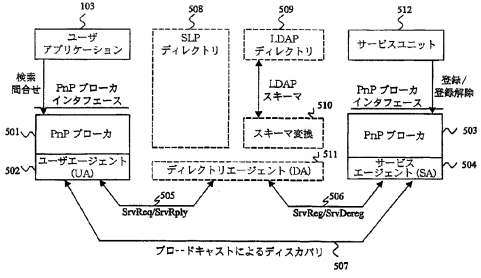
【図３】


リモート
ゲートウェイ
401
サービスユニット
402
405
403
ゲートウェイ
406
サービスユニット
PnP ブローカ
104
PnP ブローカ間
プロトコル
404
リモート
ゲートウェイ
1401
サービスユニット
1402
1405
1403
ゲートウェイ
1406
サービスユニット
PnP ブローカ
1104

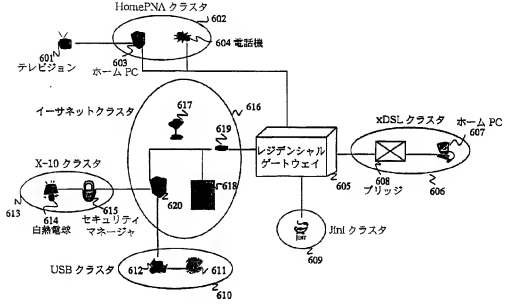
【図４】


ユーザ
アプリケーション
103
104
PnP ブローカインタフェース
302
PnP ブローカ間プロトコル
303
PnPブローカ
301
305
サービス
レジストリ
サービスディスカバリ
/アベイラビリティ
エージェント
304
サービスセッション管理エージェント
306
308
SLP ユーザ
エージェント
SLP サービス
エージェント
307
PnP ブローカ

【図５】



【図６】



フロントページの続き

(51) Int. Cl.⁷	識別記号	F I	テーマコード（参考）
Ｈ０４Ｌ　29/06		Ｈ０４Ｌ　13/00	３０５Ｂ